

Meridian Ada 4.1

Meridian ACETM User's Guide

Copyright© 1987, 1988, 1989, 1990 Meridian Software Systems, Inc. All rights reserved. No part of this manual may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without prior written permission of Meridian Software Systems, Inc. Printed in the United States of America.

The statements in this document are not intended to create any warranty, express or implied, and specifications stated herein are subject to change without notice.

Meridian Ada, Meridian ACE, Meridian-Pascal, and Meridian-C are trademarks of Meridian Software Systems, Inc.

OS/286 is a trademark of ERGO Computing Solutions.

386/ix is a trademark of INTERACTIVE Systems Corporation.

UNIX is a registered trademark of AT&T.

DECstation 3100 and ULTRIX are registered trademarks of Digital Equipment Corporation.

IBM, IBM PC, OS/2 and PS/2 are registered trademarks of International Business Machines Corporation.

Intel is a registered trademark of Intel Corporation.

Macintosh, Finder, and SANE are registered trademarks of Apple Computer Corporation.

MPW, MultiFinder and QuickDraw are trademarks of Apple Computer Corporation.

Microsoft, PC DOS, and MS-DOS are registered trademarks of Microsoft Corporation.

NFS and Sun 3 are registered trademarks of Sun Microsystems, Inc.

Except where explicitly noted, uses in this document of trade names and trademarks owned by other companies do not represent endorsement of or affiliation with Meridian Software Systems, Inc. or its products.

Table of Contents

Chapter 1 Before You Begin	1
1.1 Using this Guide	1
1.2 Getting Help	2
1.3 Exiting ACE	2
Chapter 2 Using ACE	3
2.1 Starting ACE	3
2.2 ACE Windows	3
2.3 Using Windows	4
2.3.1 The Status Line	5
2.3.2 Moving the Cursor	6
2.3.3 The DOS Window	6
2.3.4 The Execute and Review Windows	6
2.3.5 The Paste Window	7
2.3.6 The Kill Window	7
2.3.7 The Language Window	7
2.4 Using the Menu System	8
2.4.1 Selecting from Menus	8
2.4.2 Exiting from Menus	9
2.4.3 Main Menu	10
2.5 Customizing ACE	10
2.6 Making Changes Within the System	10
2.7 System Variables	11
2.8 Macros	11
2.9 Using the ACE Ada Compiler System	12
2.9.1 Developing Ada Programs	12
2.9.2 Using Ada Program Libraries	12
2.9.3 Compiling, Linking, and Running an Ada Program	13
2.10 Where to Go From Here	14
Chapter 3 ACE Tutorials	15
3.1 Preparing for the Tutorials	15
3.2 Using Files and Windows	15
3.2.1 Opening an Editing Window and Creating Files	15
3.2.2 Cut & Paste	17
3.3 Customizing ACE	18
3.3.1 Setting Editing-Related Options	18
3.3.2 Changing Key Bindings	22
3.3.3 Framing, Resizing, and Positioning a Window	24
3.4 Entering a DOS Command from ACE	25
3.5 Using ACE Macros	25
3.6 Using the ACE Language Symbol Expansion Feature	27
3.7 Compiling, Linking, and Running an Ada Program	36

Contents

3.7.1	Compiling an Ada Program	36
3.7.2	Linking an Ada Program	37
3.7.3	Running an Ada Program	37
3.8	Debugging an Ada Program	37
Chapter 4	ACE Features	41
4.1	Introduction	41
4.2	Editor Functions	41
4.3	Ada Compiler System Functions	42
4.4	Main Menu	43
4.5	File Menu	46
4.6	Edit Menu	47
4.6.1	Cursor Movement	49
4.6.2	Insert & Change Menu	51
4.6.3	Delete Menu	52
4.6.4	Find & Replace Menu	54
4.6.5	Cut & Paste Menu	58
4.6.6	Undo Menu	59
4.6.7	Macros Menu	61
4.6.8	Word Processing Menu	62
4.6.9	Language Menu	64
4.7	Windows Menu	67
4.8	Compile Menu	69
4.9	Run Menu	74
4.10	Library Menu	78
4.11	Options Menu	78
4.11.1	Edit Options (Ctrl-A)	81
4.11.2	Settings	83
4.11.3	File Options	84
4.11.4	Status	85
4.11.5	Color Menu	87
4.11.6	Set Delimiters	87
4.11.7	Write Bindings	87
4.11.8	Read Bindings	87
4.11.9	Change Bindings (Alt-2)	88
4.11.10	Write Config	88
4.11.11	Read Config	89
4.11.12	Environment Options Menu	90
4.11.13	Compile Options Menu	92
4.11.14	Link Options Menu	95
4.11.15	Make Options Menu	96
4.11.16	Run Options	97
4.11.17	Library Options	98
4.11.17.1	List Options	100
4.11.17.2	Remove Options	101
4.11.17.3	Create Options	102
4.11.17.4	Augment Options	104
4.12	Help Menu	

Appendix A Key Binding	107
Index	111

Chapter 1 Before You Begin

1.1 Using this Guide

This guide introduces you to the Meridian Ada Compiler Environment (ACE™). ACE combines the best features of an advanced editor with an interface to Meridian's Ada compiler system.

This guide has four chapters:

Chapter 1. Before You Begin

This chapter describes how to use this guide, and how to get onscreen information with ACE's online Help feature.

Chapter 2. Using ACE

This chapter contains a brief overview of ACE operations:

- Starting ACE
- Managing windows and files
- Using the menu system
- Entering commands
- Customizing ACE
- Ada programming in ACE

Chapter 3. Tutorials

Chapter 3 takes you through several tutorial sessions with ACE to illustrate the basic operations.

Note: These tutorials describe procedures as you would carry them out from the keyboard, that is, without using a mouse. However, you can use a mouse to make selections from ACE menus.

Chapter 4 ACE Reference

The final chapter is the comprehensive reference to all ACE menu selections. It is organized in the same order as the menus themselves.

As you will see, you can select all ACE functions from one or another menu. Many of them can also be selected directly with specific command keys (e.g., the function keys, Ctrl-key combinations, etc.) In general, in this guide, we identify functions by command key when available, and often mention the menu as well.

There are eight menus (File, Edit, etc.) on the main menu. All of these have submenus, and some of the submenus do as well. When referring to a function on a lower-level menu, we list them all, in sequence, separated by the symbol =>, for example:

Before You Begin

"To delete a macro, press Alt-M (Edit=>Macros)."

This tells you that Delete Macro is on the Macros menu, which is on the Edit menu, which of course is on the main menu. . .or just press Alt-M.

Note: In general, we will not bother to do this for cursor movement functions, in the tutorials.

1.2 Getting Help

You can display onscreen help at any time while using ACE. Just press Alt-H while a menu item or selection is highlighted (described later) to display information about that item.

Here's an example of the help text displayed for the "Read Config" function:

Read Config reads the options and settings from a configuration file. The default configuration file is shown. If you want a different filename, enter the desired filename.

Related topics:

Configuration File

Write Config

Press the down-arrow key to scroll through the Help text. As you do, terms on the screen for which more information is available are highlighted. Press Enter to display the information for a highlighted term.

In the example above, the highlight stops on "Configuration File" and "Write Config."

When you are finished, press ESC to "back out" of the sequence of Help screens that have been displayed, all the way to the ACE menu.

For more information on how to Help and Hyper Help, see section 4.12. This section also provides the information needed to access the online versions of the Ada Language Reference Manual and DOS Environment and Ada Utility Library documentation.

You can also select Help from the ACE main menu. This is described in more detail in the next chapter.

1.3 Exiting ACE

You can exit ACE at any time by using the Ctrl-Z (Exit) or Alt-Z (Abort) keys. Ctrl-Z exits ACE saving your file changes. Alt-Z exits ACE without saving your changes.

Chapter 2 Using ACE

ACE combines the features of a powerful multiwindow editor with the Meridian Ada Compiler System in a versatile, easy-to-use, menu-controlled environment. This chapter provides a brief explanation of how to use the environment. Reading this chapter will prepare you for the tutorials in the following chapter.

2.1 Starting ACE

Before you start ACE, make sure you have installed the software according to the instructions in the *Getting Started* manual.

If you have done that, your next step is to change to a directory (using the DOS command **CD**) that contains the files you intend to edit (or where you would like to store them, if you haven't created them yet). If you defined the **ACEDIR** variable properly (as described in the installation instructions), and put the **ada\bin** directory in your path, then you can start ACE from any directory you like.

If the system is installed and you are in the directory you want, simply type **ACE** and press **ENTER**. In a moment or two, you will see this screen:

[ACE startup screen; no file]

The number "1" in the upper left corner tells you that you are currently in window number 1. (With ACE, you can edit in up to ten windows, and there are six others with specialized functions. Windows are covered in the next section of this chapter.)

Below the window is the *status line*, which tells you the current location (line and column) of the cursor and various other details that we'll come back to. (See Chapter 4 for more comprehensive information on this and other ACE features).

The box in the middle of the window contains the title and copyright information, and the important fact that you can get onscreen help about virtually any aspect of ACE, at any time, by pressing **Alt-H**. As soon as you press a key, this box will disappear, and the status line will be temporarily replaced by this prompt:

Filename:

Here, ACE is asking for the name of a file to edit. Type a legal DOS file name, and it will be loaded (if it exists in the current directory) or created.

As you will see in the next section, you must always have a file name associated with any window you are using. When you exit ACE, the current state of the system is saved, and reloaded automatically the next time you start ACE from that directory. In that case, the file is present when you start, and you are not prompted for a file name.

2.2 ACE Windows

A window is sort of a "virtual screen." In ACE, you can use as many as ten windows for editing files. You might have a single file in all ten, with different parts of the file displayed in each, so that you can refer to any

Using ACE

of them with a single keystroke. Or you might load different files into each window, and switch from one to another.

Note that, whenever an editing window is displayed, it must have a file name associated with it. There is no such thing, in ACE, as an unnamed window.

In addition to the ten editing windows, ACE has six specialized windows:

Window	Function
DOS	For entering DOS commands without exiting ACE.
Execute	Displays compiled Ada source code; used with the Review window for fixing compilation errors.
Review	Displays error messages for the current compilation.
Paste*	Displays the text that has most recently been copied or cut, and which will be pasted into the current window if you select Paste. You can add, delete, or edit this text, but you cannot cut and paste within this window.
Kill*	Displays the last few lines of deleted text, which you can restore using the Undelete Line and Yank Line functions.
Language*	This window contains the Ada language syntax rules file. It is used by the language symbol expansion functions. You should not change this file unless you are very familiar with its purpose.

*Under normal circumstances these windows can be ignored. These windows are implicitly used by ACE.

2.3 Using Windows

When you start ACE, it automatically reloads the state of the system as it existed the last time you started it from that directory, if any. Therefore:

- The first time you start ACE from a given directory, it prompts you for a file name to create or retrieve.
- If you have previously started ACE from a given directory, it loads the same file(s) into the same window(s) as when you last exited ACE.

There are several ways to switch from one window to another, using the Windows menu (menus are described in the next section) or key combinations:

Function	Key	Description
Select	Alt-F6	Displays a list of windows and the file (if any) each contains. Use the up and down arrow keys to highlight the window you want and press ENTER.
Switch to Window	Alt-F5	Prompts for a window number to switch to and, if that window is blank, a file name to retrieve or create.
Close Window	Shift-F2	Disassociates the file from the current window and switches to the next most recently displayed window. If no other window has a file in it, prompts you for the name of a file to open.

In the first function, Select, if the new window has no file name associated with it, the file in the current window is opened in the new window.

You can also move from one window to another by pressing **Ctrl-F<window number>**. That is, **Ctrl-F1** moves you to window 1, and so on. As with **Select**, the file in the current window moves with you.

Note that editing a file in one window affects copies of that file in other windows; however, the display is not affected. That is, as you move from one window to the next, the part of a file displayed in a window, and the cursor position, does not change. You'll see some ways to take advantage of this in the tutorials in Chapter 3.

2.3.1 The Status Line

At the bottom of the screen, the status line shows:

- The window number (or name)
- The current cursor position
- The edit settings for all windows (covered in "Customizing ACE," later in this chapter)
- The file name associated with the current window
- The current size of this file, and of the Kill and Paste buffers

Here is a sample status line, with all edit settings (the letters in <angle brackets>) turned on:

```
[1] Line 1 Col 1 <DBMIWSTN> K:24 P: 1 TEST.ADA 86
```

From left to right, the indicators are:

[1] The number of the currently active window. In the case of a specialized window, such as the Kill window, the window name appears here instead of a number.

Line #	Current line number (cursor location)
Col #	Current column number (cursor location)
<D>	Line Drawing mode is on
	Bullet mode is on
<M>	Keystroke macro recording is on
<I>	Insert mode is on
<W>	Word wrap is on
<S>	Case-sensitive search is on
<T>	Auto-Tab is on
<N>	Auto-Insert is on
K: #	Current size (# bytes) of the Kill Buffer
P: #	Current size (# bytes) of the Paste Buffer
TEST.ADA	Current file name
86	Size (in bytes) of the current file.

You can change the edit settings with the Options menu. For more information, refer to the Options menu items in Chapter 4, and "Customizing ACE," later in this chapter.

2.3.2 Moving the Cursor

Cursor control features in ACE are initially bound to standard cursor keys as follows:

Up/down a row	↑/↓
Left/right a column	←/→
Left/right a word	Ctrl-←/→
Left side of screen	Ctrl-Home
Right side of screen	Ctrl-End
End of current line	Ctrl-E
Upper left of screen	Home
Lower right right of screen	End
Up/down a screen	PgUp/PgDn
Start of file	Ctrl-PgUp
End of file	Ctrl-PgDn
Goto line number n	Ctrl-G n

Most of these are self-explanatory. All of them appear on the Edit=>Cursor Movement menu. If you want to change them to correspond to an editor that you are more familiar with, you can rebind them as you see fit. See "Customizing ACE," later in this chapter.

2.3.3 The DOS Window

Use the DOS window to enter DOS commands without exiting ACE. You can enter any valid DOS command (e.g. DIR). The last 2500 bytes of output are saved in this window. To return to ACE, press Esc.

See "Operating System Services", in Chapter 4, for a typical DOS Window procedure. In that example, the DOS command **CHKDSK** is entered to determine the current status of the PC hard disk.

2.3.4 The Execute and Review Windows

The Execute Window is typically used to display Ada program source code that has been compiled and is now being reviewed for errors.

The Review Window contains the error messages (up to 1000 lines) associated with the current compilation.

When you compile a program from within ACE (described later), the Execute and Review Windows are displayed together, so that you can identify lines of code that caused compilation error messages.

For example, the following Execute-and-Review window display might appear after a compilation failure:

```
with e;
use direct_io;
separate ( sample )
<<proper_body>>
```

```
sample.ada, 2: undefined package in use clause "direct_io" [LRM 10.1/1]
sample.ada, 3: subunit ancestor is not in library "sample" [LRM 10.2]
sample.ada, 4: illegal compilation unit [LRM 10.1/2]
sample.ada, 6: semicolon expected [LRM 3.9/2]
6 lines compiled.
4 errors detected.
```

The Execute window, on top, shows the source code, while the Review window, underneath, identifies each error, citing the appropriate Ada Language Reference Manual (LRM) section.

In the actual Review window, one error line is highlighted, corresponding to the code line marked by the cursor in the Execute window. The Review window also summarizes the compilation results, indicating how many lines were compiled and the total number of errors.

2.3.5 The Paste Window

The Paste Window lets you edit and review previously copied or cut text (stored in the Paste Buffer). The Edit=>Cut & Paste menu includes functions for:

- Marking text to cut or copy
- Cutting and copying text to replace (or be appended to) any text already in the Paste Window
- Displaying the Paste Window

You can also reach the Paste Window via **Windows=>Select** (Alt-F6).

You can edit the contents of the Paste Window by adding, deleting, or changing text before it is pasted into a file. However, you cannot use the Cut or Paste functions on text inside the Paste Window.

The contents of the Paste Window are saved on disk, so virtually any amount of cut or copied text can be preserved.

2.3.6 The Kill Window

The Kill Window is used to review deleted text (stored in the Kill Buffer). You can undelete or yank text from this window, and paste them into an edit window.

You can display the Kill Window with **Windows=>Select** (Alt-F6).

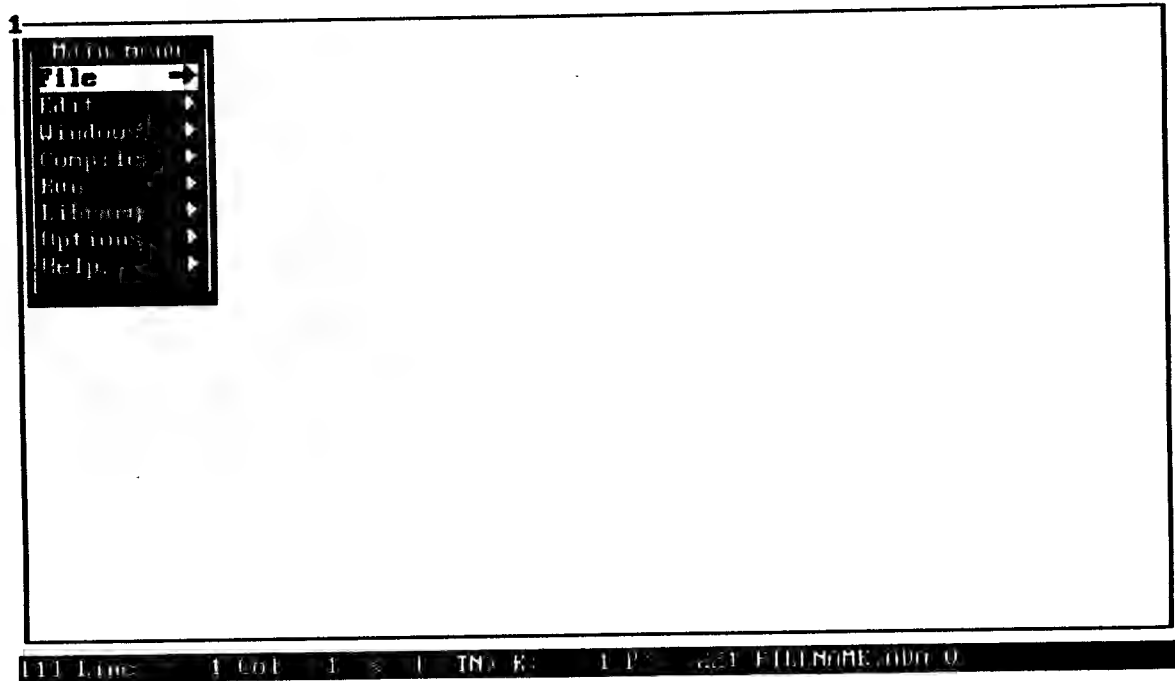
The Kill Buffer holds up to 650 characters, or about 25 lines. After that limit, later deletions displace earlier ones (first in, first out).

2.3.7 The Language Window

This window contains the Ada language syntax rules file. It is used by the language symbol expansion functions. You should not change this file unless you are very familiar with its purpose.

2.4 Using the Menu System

The most frequently used ACE functions are assigned to various command keys. However, all ACE functions can be accessed through the menu system. From any ACE window, press ESC to display the main menu:



The main menu always appears at the current cursor position.

2.4.1 Selecting from Menus

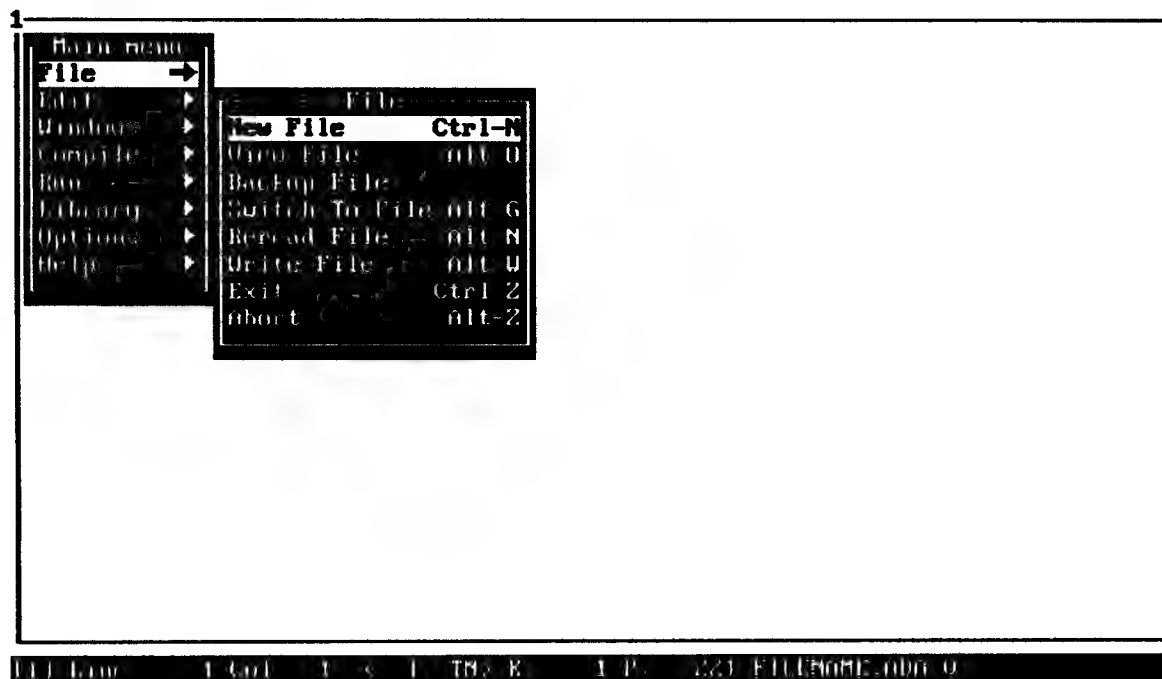
On any ACE menu, you select an item by highlighting that item and pressing ENTER. To highlight an item:

- Move the highlight to the item with the up and down arrow keys. . .

OR

- Press the first letter of the item. The highlight will move to the first item that begins with that letter. If that is not the item you want, press that letter again to move the highlight to the next item beginning with that letter.

Whenever a menu item is followed by the symbol =>, selecting that item displays a submenu. As you can see in the illustration above, all main menu items display other menus when selected. For example, here is the menu displayed when you select "File":



When a key is specified to the right of an item, it indicates that you can select that item directly from an ACE window without using the menu system, by pressing that key. (Of course, you can still select those items from the menu as described above.)

As you can see, none of the File Menu items displays a submenu; and there are command keys assigned to all of the File items except for "Backup File."

With ACE, you can change the key bindings at any time—that is, assign command keys to functions that don't have them and change existing key bindings. The menu displays reflect changes to key bindings.

Try this for yourself in the tutorial section of this guide, or read the details under "Change Bindings" (items menu) in Chapter 4.

2.4.2 Exiting from Menus

In most cases, completing a logical sequence of menu selections returns you automatically to the ACE window you started from. For example, you might select Edit from the main menu; Delete from the Edit menu; and Delete Line from the Delete menu. At that point, the line would be deleted and you would be back in your file.

However, you can always "back out" of the menu sequence, one menu at a time, by pressing **ESC**.

2.4.3 Main Menu

The main menu is displayed when you press ESC from any ACE window:

Main Menu	
File	=>
Edit	=>
Windows	=>
Compile	=>
Run	=>
Library	=>
Options	=>
Help	=>

The main menu is always displayed just to the right of the current cursor position. If the cursor is in Line 1, Column 1, it appears at the upper left corner of the screen.

All main menu selections display lower-level menus. The purposes of these menus are summarized below. For more details on these menus, and on the items in still lower-level menus, see Chapter 4, "ACE Functions."

Menu	Function
File	File manipulation (create, retrieve, save, etc.); abort editing or exit ACE.
Edit	Nine lower-level menus for moving the cursor, deleting text, cut and paste, macros, etc.
Windows	Managing windows: moving from one to another, changing size or position, closing, etc.
Compile	Compile and link a file, run the Make program, and get information about the current compilation.
Run	Run and debug programs, issue DOS commands.
Library	Manage Ada program libraries.
Options	Review and change various aspects of the ACE environment: editing defaults, key bindings, configuration, compile and link items, colors, etc.
Help	Display onscreen Help or HyperHelp.

2.5 Customizing ACE

In general, to change the way ACE operates, you use menus or command keys within ACE. One exception: to determine which directories and files ACE uses on startup, you can change DOS *environment variables* from the DOS prompt.

More complete information on techniques for customizing ACE is available in Chapter 4; and several of these procedures are demonstrated in the tutorials in Chapter 3.

2.6 Making Changes Within the System

To change the size or position of the current window, use the Windows menu. Most other changes involve using various submenus of the Options Menu: Edit Options, Settings, File Options, Status, Color, Change Bindings, Environment Options, and others.

Some changes—for example, to windows—are automatically carried over to future sessions. In other cases, such as changes to key bindings, you are asked whether you want to save the changes in a configuration file. If you answer No, the changes are in effect for the current session only.

If you answer Yes, a path and file name for the configuration file are suggested, and you may accept or edit them. Thus you can have different configuration files in different directories. Configuration files in the directory that you start ACE from are the ones used in that session.

You can also read in a new configuration file while you are using ACE, putting a complete set of changes into effect at any time. (The Read Config item is on the items menu.)

To return to the defaults within an existing directory, just delete the default configuration files in that directory. (The default names for configuration files are **ACE.CFG** for items and settings, and **ACE.KEY** for key bindings; but you can specify other names when you save these files.) The next time you start ACE from that directory, it will have the default settings and bindings.

For more information about specific changes, see the entries for the Options Menu submenus, and the Windows Menu, in Chapter 4.

2.7 System Variables

From DOS, you can define or change system-variables to use different directories and files within ACE.

System variables for ACE are defined in your **AUTOEXEC.BAT** file. They include the following:

Variable	Required?	Description
ACEDIR	Yes	Identifies the directory where ACE is installed.
ACECFG	No	Identifies the configuration file, which defines all ACE items and settings. See “Customizing items and Settings”, below.
ACEKEY	No	Identifies the key-binding file. See “Customizing Key Bindings,” below.
ACEMAC	No	Identifies the keystroke macro file.
ACEPST	No	Identifies the paste buffer file. If this variable is not defined, ACE uses the file ACE.PST in the directory identified by ACEDIR .
MCP SWAP	No	Sets the file for ACE to swap out to when another program (e.g., the compiler) is loaded.

To change a variable for the current session, use the DOS command SET before running ACE. For example, at the DOS prompt, type:

```
SET ACECFG=<name of new configuration file>
```

To change any variable across sessions, put the appropriate SET commands in your **AUTOEXEC.BAT** file, as described in the installation instructions, in the Getting Started manual.

2.8 Macros

There is one other facility for customizing ACE that should be mentioned here: creating macros.

A macro is a sequence of keystrokes-text or command keys, or both-that you can save and then execute in various ways.

In ACE, you can define a macro with Ctrl-V and execute it with Ctrl-X. If you want to keep that macro for use in later sessions, you can name it and save it in a macro file. When you want to use the macro(s) in that file, you can read it in.

Finally, you can bind a macro to an unassigned command key. (Command key assignments are listed in Appendix Appendix A.)

2.9 Using the ACE Ada Compiler System

From ACE, you can manage Ada program libraries and develop, compile, link, and execute Ada programs. Library management and Ada program development are explained briefly in this section, and described with more detail in Chapter 4.

2.9.1 Developing Ada Programs

Ada programs are developed in ACE by writing code manually, or by using the interactive Language Symbol Expansion functions.

You can develop Ada programs manually in any ACE editing window. After writing the program, you compile the file, and link the program.

To use the interactive, system prompt–user response mode with ACE Language Symbol Expansion functions, you start the same way, by naming the file in any editing window. Then use the Edit => Language menu (or command keys) to start and manage the interactive mode.

You can try this in the tutorial section, or refer to the description in Chapter 4.

2.9.2 Using Ada Program Libraries

An Ada program library is a collection of files (or database) used by the Ada Compilation System to keep track of the results of compiling Ada source files and their associated compilation units. Before you compile your first Ada source file in a given directory you must create the initial program library. This can be accomplished with the Library => Create New Library function.

Typically a given Ada program library contains all the application–specific compilation units that make up a particular application program or it contains a collection of re–usable units which provide a service of general applicability to many different programs. To make it easy to manage these common, general–purpose compilation units, a given Ada program library can reference the units defined in one or more other libraries. Such references are usually referred to as library links. The Meridian Ada Compiler comes with several pre–defined (that is, pre–compiled) Ada program libraries. Others may be purchased separately. Still others you may develop on your own.

By default, the Library => Create New Library function adds library links to the following pre–defined Meridian Ada Libraries to the new library. These defaults can be changed to suit your own needs (see the Options => Library Options => Create Options Menu). Library links can also be modified after the initial library is created (see the Library Menu).

- Standard Program Library
- Ada Utility Library
- DOS Environment Library

Standard Program Library

The Standard Program Library, for the most part, contains those compilation units which are required by the LRM. They include such packages as `calendar`, `direct_io`, `machine_code`, `sequential_io`,

system, and **text_io**. Note that the LRM defined package which is actually called **Standard** is defined implicitly by the Ada compiler and is not contained in this library.

See the *Meridian Ada Compiler User's Guide* for more information about the Standard Program Library.

Ada Utility Library

The Ada Utility Library contains miscellaneous Ada packages of general applicability.

The Ada Utility Library packages enable a program to:

- Access command line arguments (**Package arg**)
- Declare array objects larger than 64K (**Generic Package array_object**)
- Declare array types larger than 64K (**Generic Package array_type**)
- Perform inline bit-level operations (**Package bit_ops**)
- Use floating point transcendental functions (**Package math_lib**)
- Flush output text files (**Package spio**)
- Perform byte peek and poke operations (**Package spy**)
- Handle text according to LRM 7.6 requirements (**Package text_handler**).

See the *Meridian Ada Compiler User's Guide* for more information about the Ada Utility Library packages.

DOS Environment Library

The DOS Environment Library contains Ada packages that enable your program to interface directly with the operating system of your PC (PC- or MS-DOS). The library allows you to use most DOS system calls, screen management features, and other BIOS functions.

Individual DOS Environment Library packages provide or enable:

- Graphics and character display routines (**Packages box, cursor, tty, and video**)
- System time determination and setting procedures (**Package time**)
- DOS file operations (**Package file_io**)
- Disk operations (**Packages disk and absolute_disk**)
- Program control and execution procedures (**Package program_control**)
- Memory area allocation and deallocation procedures (**Package memory**).

See the *Meridian DOS Environment Library User's Guide* for more information about the DOS Environment Library packages.

2.9.3 Compiling, Linking, and Running an Ada Program

The ACE Options=>Compile Options menu lists many compile options that you can turn On or Off before you compile your program. These options include:

- Generate 80286 instructions
- Perform local optimization

- Generate debugging information
- Generate assembly code
- Generate listing file

...and many others. For more information about compile options, see the *Meridian Ada Compiler User's Guide*. The important thing to remember is that you must set these options *before* you compile the program.

Similarly, the Options => Link Options, Make Options, and Run Options menus should be set before you take these steps. Information on them is also available in the *Meridian Ada Compiler User's Guide*.

After you compile a program, any errors found are displayed on a split screen. The Execute Window, at the top, displays your source code. The cursor is on the line containing the first error.

The Review Window, at the bottom, contains messages identifying each error with an indication of the nature of the mistake and a reference to the relevant LRM section. The highlighted description line corresponds to the cursor-marked line in the Execute Window.

You can move to subsequent errors in the compilation (Run => Next Error) or go back to previous ones (Run => Previous Error).

After you correct the errors in the Execute window, you recompile the file. After an error-free compilation, you can link the program (Compile => Link Program), and execute it (Run => Run Program).

2.10 Where to Go From Here

We suggest that you go through the tutorials in Chapter 3 to get some hands-on experience with ACE. However, if you feel that you know enough now to get started with your own work, you can do so, using Chapter 4 as a reference to specific ACE features.

BUT...if you intend to use the Language Symbol Expansion feature, we urge you to go through the section of the tutorial that covers that feature first. In that tutorial, you develop a short Ada program.

Chapter 3 ACE Tutorials

ACE provides flexible tools for editing, compiling, linking, and running Ada programs. This chapter gives you a chance to try out some of these tools in guided, hands-on tutorials:

- Using files and windows
- Customizing ACE
- Entering a DOS command from ACE
- Using ACE macros
- Using the ACE Language Symbol Expansion Feature
- Compiling, linking, and running an Ada program
- Debugging an Ada program

All keystrokes to complete the tutorials are provided in the text, along with explanations (where necessary) and the menu alternatives, if any. (We won't bother to note the menu alternatives for cursor-movement keys.)

Note: The steps in these tutorials assume that your ACE system has been installed according to the directions in *Getting Started*, and that you are using the default key bindings with which the system was shipped. If you have changed any of these bindings, substitute the new keystrokes or use the menu indicated to select the feature.

3.1 Preparing for the Tutorials

To begin, we suggest that you create a tutorial directory and always start ACE from there when doing the tutorials. This will keep the tutorial files (data and configuration files) segregated from your normal work area.

1. Create the directory `\ACETUT`. At the DOS prompt, type:

```
md \acetut <ENTER>
```

Note: Text in <angle brackets> describes (rather than specifies) your entry. Thus <ENTER> means "Press ENTER," not "Type the characters <ENTER>."

2. Change to the tutorial directory:

```
cd \acetut <ENTER>
```

Make sure you are in this directory whenever you start ACE for the purpose of doing these tutorials.

3.2 Using Files and Windows

This tutorial demonstrates several ways to create and retrieve files, and view them in different windows.

3.2.1 Opening an Editing Window and Creating Files

1. Start ACE. At the DOS prompt, type:

ACE <ENTER>

A blank window appears with the ACE logo.

2. Press ENTER (or any character key) to clear the logo from the screen. This prompt appears:

Filename:

3. Type the file name and press ENTER.

file1.ad <ENTER>

This prompt appears:

File does not exist. Create?
Yes
No

4. "No" is highlighted. Select "Yes" by moving the highlight there and pressing ENTER.

Y <ENTER>

To move the highlight, press the first character of the option you want, or use the up and down arrow keys.

Window 1, an Editing Window, now contains the blank file, **file1.ad**, as you can tell by the file name on the right side of the status line.

5. Type some text into this file.

This text is part of file1.ad.

6. Move to Window 2, taking this file with you.

<Ctrl-F2>

Window 2 replaces Window 1, with the same text in view. The file exists in both windows. In Window 2, the cursor is under the first character of the text.

7. Move the cursor to the end of the line and add some text to the file.

<Ctrl-E>

<ENTER>

<ENTER>

<ENTER>

This was added in Window 2.

8. Switch back to Window 1.

<Ctrl-F1>

The new text also appears in Window 1, but notice that the cursor is still where it was when you last occupied that window.

9. Move to Window 6.

<Ctrl-F6>

This window occupies the bottom half of the screen, so you can see the text in both Window 1 and Window 6.

10. Give the command to replace `file1.ad` with a new text file in the current window.

`<Ctrl-N>`

ACE quickly saves the current contents of this window and prompts you for the new filename:

[6] Filename:

11. Identify the new file.

`file2.ad` `<ENTER>`

File does not exist. Create?
Yes
No

12. Highlight Yes and press ENTER.

Window 6 now contains `file2.ad`, which is blank.

13. Display the Select Windows menu.

`<Alt-F6>`

Menu: Windows=>Select

This shows the contents of all the Editing Windows, as well as the special windows. The cursor is at Window 6, which is blinking, to indicate that it is the current window.

We'll use the DOS Window in a later tutorial.

14. Select Window 8.

`8` `<ENTER>`

You can also move the cursor with up and down arrow keys.

15. Move directly to Window 1.

`<Ctrl-F1>`

As soon as you move to Window 1, it expands to fill the screen. This is its default size. A later tutorial takes you through the steps for changing window size and position.

3.2.2 Cut & Paste

The next few steps show you how to paste text from one file/window into another. First, notice the value after the P: on the status line. It should be 0.

1. Move the cursor to the Home position (top left).

`<Home>`

2. Mark the text in this file.

`<F1>` `<End>`

3. Copy the marked text.

`<F3>`

Menu: Edit=>Cut & Paste=>Copy

The value after the P: in the status line should now be 103 (if it's slightly different, don't worry about it). This indicates the size of the Paste buffer, which now contains the text you copied.

4. Move to Window 8.

<Ctrl-F8>

5. Paste the copied text.

<F7>

Menu: Edit=>Cut & Paste=>Paste

The text you copied from Window 1 has been pasted here from the Paste buffer.

That ends this portion of the tutorial. If you like, experiment with entering text and moving among the Editing Windows. Try some other editing features, using the information in Chapter 4. (Don't worry about making changes to the files; the exact contents are unimportant.)

The next tutorial takes you through the steps for customizing certain aspects of ACE.

If you want to exit ACE before starting the next tutorial, do so by pressing Ctrl-Z (**File=>Exit**). This saves the current state of the system before quitting.

To restart ACE, follow the same steps as before: change to this directory, type **ACE** and press ENTER. It will restart with the same files in the same windows as when you exited.

3.3 Customizing ACE

This tutorial demonstrates several ways to customize the ACE editing environment, including adjusting Editing Windows.

These are the options you'll change:

- Word wrap
- Justify
- Tab size
- Break long lines
- Status line
- Key binding for Delete Word
- Window size, position, and colors

Note that the new options you set will be in effect only when you start ACE from the current directory. This is one reason we suggest you create a new directory for these tutorials.

To begin, restart ACE (if necessary) by typing the command **ACE** from DOS in the tutorial directory.

3.3.1 Setting Editing-Related Options

1. In the Editing Window, access the Edit Options menu.

<Ctrl-A>

Menu: Options=>Edit Options

The Edit Options menu appears.

2. Turn on Word wrap. Highlight "Word wrap" with arrow keys or by pressing "W".

<ENTER>

A check mark appears to the left of "Word wrap."

3. Turn on Justify in the same way. Highlight "Justify" with arrow keys, or by pressing "J".

<ENTER>

Note: The following options are already turned on, by default:

- Insert characters
- Auto-tab
- Auto-insert
- Trim spaces
- Tab compress/expand
- Retain Log Across Sessions
- Save/Restore
- Retain Paste Buffer Across Sessions
 - Shift By Tab Stops
 - Automatic Language Expansion

For more information on any of these options, see the entry for this menu in Chapter 4.

4. Leave Edit Options menu.

<ESC>

This prompt appears:

```
Write Changed Settings To Config File?
Yes
No
```

5. Leave Yes highlighted and press ENTER.

The next prompt is:

```
Write config file [default=ACE.CFG]:
```

You could press ESC to cancel writing out the changes, or type in another path or filename here. If you want the changes to apply no matter which directory ACE is started from, then write them out to the file **ACE.CFG** in the ACE directory (the directory given by the **ACEDIR** system variable).

6. Press ENTER.

You return to the Editing Window. The new options have been saved in the configuration file named in the prompt. They will apply whenever you start ACE from this directory.

7. Access the Settings menu.

<Alt-A>

Menu: Options=>Settings

The Settings menu appears:

8. Select Tab size.

T <ENTER>

The Settings menu disappears and you are prompted (at the bottom of the screen):

Tab [8]

9. Change the Tab size to 12.

12 <ENTER>

This prompt appears:

**Write Changed Settings To Config File?
Yes
No**

10. Leave Yes highlighted and press ENTER.

The next prompt:

Write config file [default=<directory>\ACE.CFG] :

11. Accept the path and filename shown.

<ENTER>

You return to the Editing Window. The new tab size has been recorded in the configuration file, and will apply whenever you start ACE from this directory.

12. Access the File Options menu.

<Options=>File Options>

There is no command key bound to this menu.

The File Options menu appears.

13. Turn on "Break Long Lines" and exit this menu.

B <ENTER> <ESC>

These file options are already turned on, by default:

- Delete 0 Length Files
- Backup
- Retain Backup Across Sessions

When you press ESC, you are prompted:

Write Changed Settings To Config File?

Yes

No

14. Accept the default, "Yes."

<ENTER>

At the next prompt:

Write config file [default=<directory>\ACE.CFG] :

15. Press ENTER.

You return to the Editing Window. The new option has been recorded in the configuration file, and will apply whenever you start ACE from this directory.

16. Access the Status menu.

<Options=>Status>

There is no command key bound to this function.

The Status menu appears.

17. Select Memory Available and exit this menu.

M <ENTER> <ESC>

All other options on this menu are already turned on, by default. With Memory Available turned on, the amount of memory available will display on the status line.

The Settings menu disappears and you are prompted:

Write Changed Status Settings To Config File?

Yes

No

18. Leave Yes highlighted and press ENTER.

The next prompt:

Write config file [default=<directory>\ACE.CFG] :

19. Accept the path and filename shown.

<ENTER>

You return to the Editing Window. The status line now displays the amount of memory available, after the indicator "M:". This will be the case whenever you start ACE from this directory.

IF YOU HAVE A MONOCHROME MONITOR, we suggest that you skip (or just read through) the rest of this part of the tutorial on setting color

20. Access the Window Color menu.

<Alt-F10>

Menu: Options=>Color=>Window Color

The Select Color menu appears.

This menu shows the current foreground and background colors for the current window,

Note that you use this same display to set color for all Color menu selections except "Color/B&W" (which toggles between the two kinds of display) That is, if you select Status Color, the same display appears, showing the current foreground and background colors for the Status Line.

On this menu:

- Use the up and down arrow keys to select Foreground or Background
- Use the left/right arrow keys to cycle through the colors available.

21. Highlight Foreground and select the foreground color (e.g., pale blue, white, etc.) using the arrow keys.

The window characters will be in the color you select.

22. Highlight Background and select the background color using the arrow keys.

The window background will be in the color you select.

23. Save the color setting when you are satisfied with it.

<ENTER>

The usual prompt appears:

```
Write Changed Color To Config File?
Yes
No
```

24. Accept the default, "Yes."

<ENTER>

At the next prompt:

```
Write config file [default=<directory>\ACE.CFG] :
```

25. Press ENTER.

You return to the Editing Window with the new colors in effect, as they will be whenever you start ACE from this directory.

This is a good place to take a break, if you like.

3.3.2 Changing Key Bindings

If you exited ACE after the last part of the tutorial, make sure you are in the tutorial directory (if you created one) and give the command ACE to restart.

The next few steps show you how to assign a command key to a function on a menu. You'll assign the Delete Word function to Ctrl-F. (By default, Ctrl-F is bound to the Find function. For a handy list of ACE functions, by function and also by key, see Appendix Appendix A of this guide.)

1. To begin, check the Delete menu to make sure that Delete Word is not currently bound to a key.

Edit=>Delete

There is no command key listed to the right of Delete Word.

2. Return to the editing screen.

<ESC> <ESC> <ESC>

3. Access the Change Bindings list.

<Alt-2>

Menu: Options=>Change Bindings

The list appears, with directions beneath it:

Esc-cancel; Enter-Bind function; [etc]

4. Move the highlight to Delete Word. Move the highlight with PgDn, arrow keys, or first letter of entry (D).
5. With Delete Word highlighted, select it.

<ENTER>

This message appears at the bottom of the screen:

Bind Delete Word to: _

6. Press Ctrl-F.

In a moment, the Change Bindings menu reappears, with Ctrl-F to the right of Delete Word.

At this point, you could select another command to bind to a key.

7. Exit this screen.

<ESC>

This prompt screen appears:

Write Bindings To File?
Yes
No

8. Select Yes.

<ENTER>

The prompt becomes:

Write key binding file [defatult=ACE.KEY]:_

As usual, unless you enter a new path or filename, the change will be saved in the file named here, in the current directory.

9. Accept the default file.

<ENTER>

<Alt-N> BOX <ENTER> Y <ENTER>

Menu: File=>New File

2. Turn on the Define Macro function.

<Ctrl-V>

Menu: Edit=>Macros=>Define Macro

There is only one change on the screen: the status line displays the letter "M" to indicate that Macro Define is on. For example:

[1] Line 1 Col 1 <MI TN> K: 1 P: 0 <BOX> 0

(See "Using ACE Windows", in Chapter 2, for more information about the Status Line.)

Whatever you type now, becomes part of the macro until you end Macro Define.

3. Turn on the Line Drawing function.

<Alt-Q>

Menu: Insert & Change=>Line Drawing

Again, the only change is in the status line: it displays the letter "D" to indicate that Line Drawing is on. For example:

[1] Line 1 Col 1 <DMI TN> K: 1 P: 0 <BOX> 0

4. Construct a box using the arrow keys. Use the arrow keys to move the cursor through a rectangle (see below).

Just before the cursor reaches the upper left corner of the figure, the line will come within half a line of the top; press -> (assuming you've been drawing clockwise) once at this point to make the lines meet, and the corner will be square.

If the box doesn't come out just right, don't worry about it. Any figure will do, since it is for demonstration purposes only.

5. Turn off Line Drawing.

<Alt-Q>

Menu: Insert & Change=>Line Drawing

The letter "D" disappears from the Status Line at the bottom of the screen.

6. Turn off Define Macro.

<Ctrl-V>

Menu: Edit=>Macros=>Define Macro

The letter "M" disappears from the Status Line at the bottom of the screen.

The macro you just defined has no name, except the "current macro." There can be only one current macro at a time-if you define another macro, it replaces this one. You can, however, name and save any number of macros. See the section on Macros in Chapter 4 for more information.

7. Move the cursor to a point in the document where you would like to redraw the box and execute the macro.

<Cursor keys> <Ctrl-X>

Menu: Edit=>Macros=>Execute Macro

The steps that you took while Macro Define was on are repeated, and the same figure is redrawn at the new location.

Note that Ctrl-X executes the current macro, i.e., the last macro defined. If you define a new macro, or exit ACE, without naming and saving this one, the current macro is lost.

3.6 Using the ACE Language Symbol Expansion Feature

The ACE Language Symbol Expansion Feature is a tool for writing Ada programs. Basically, it inserts general terms for required parts of a program and lets you expand them into a list of more specific terms from which you can choose.

In this tutorial, you use Language Expansion to write the following short program:

```
with text_io;
use text_io;
procedure hello is
begin
  put_line("hello");
end hello;
```

In the tutorial that follows, you compile, link, and run the program, which simply displays:

```
hello
```

Note: If you decide to skip this tutorial, you can simply type in the program to use it in the next tutorial.

Although the program is short, the tutorial is relatively lengthy. Every possible expansion product is shown for each expandable symbol.

For example, if a <<library_or_secondary_unit>> is to be expanded, a submenu opens to let you choose one of these expansion products:

```
<<subprogram_body>>
<<subprogram_declaration>>
<<package_declaration>>
<<package_body>>
<<generic_declaration>>
<<generic_instantiation>>
```

Like other Language Expansion choice menus, this one is based on the LRM (e.g., LRM 10.1 "Compilation Units – Library Units").

If you exited ACE after the last part of the tutorial, make sure you are in the tutorial directory (if you created one) and give the command ACE to restart.

1. Create the file `hello.adb`.

```
<Ctrl-N>
```

```
Menu: File=>New File
```

```
hello.adb <ENTER>
```

```
Y <ENTER>
```

2. Make sure that ACE is set for logging and automatic language expansion.

```
<Ctrl-A>
```

8. This time, select Expand.

The screen shows the expansion and a submenu:

```
with text_io;
use <<unit_simple_name>>, <<unit_simple_names>>;
<<context_clauses>>
<<library_or_secondary_unit>>
    <<unit_simple_name>>
        <<INPUT_package_identifier?>>
        <<simple_name>>
        ascii
        calendar
        direct_io
        io_exceptions
        machine_code
        sequential_io
        standard
        system
        text_io
        unchecked_conversion
        unchecked_deallocation
        BACKUP TO PREVIOUS SYMBOL
```

9. Select text_io.

The screen shows the expansion and prompts you for the next optional item:

```
with text_io;
with text_io<, <<unit_simple_names>>;
<<context_clauses>>
<<library_or_secondary_unit>>
```

<, <<unit_simple_names>>;: E(xpand), D(elete), B(ackup) or Q(uit)

10. Select Delete.

The screen shows the deletion and prompts you for the next optional expandable symbol:

```
with text_io;
use text_io;
<<context_clauses>>
<<library_or_secondary_unit>>
```

<<context_clauses>>;: E(xpand), D(elete), B(ackup) or Q(uit)

11. Select Delete.

The screen shows the deletion and displays a choice menu for the next expandable symbol:

```
with text_io;
use text_io;
<<library_or_secondary_unit>>
```

```

<<library_or_secondary_unit>>
<<subprogram_body>>
<<subprogram_declaration>>
<<package_declaration>>
<<package_body>>
<<generic_declaration>>
<<generic_instantiation>>
<<subunit>>

```

BACKUP TO PREVIOUS SYMBOL

12. Select <<subprogram_body>>.

The screen shows the expansion and displays a choice menu for the next expandable symbol:

```

with text_io;
use text_io;
<<subprogram_body>>
<<subprogram_body>>
<<procedure_body>>
<<function_body>>

```

BACKUP TO PREVIOUS SYMBOL

13. Select <<procedure_body>>.

You are now prompted for the procedure's name:

Input procedure identifier:

14. Type **hello** and press ENTER.

The screen shows your substitution and prompts you for the next option:

```

with text_io;
use text_io;
procedure hello<<formal_part>>> is
    <<<declarative_part>>>
begin
    <<sequence_of_statements>>
    <<<exception_handler_section>>>
end hello;

```

<<<formal_part>>>: E(xpand), D(etele), B(ackup) or Q(uit)

15. Select Delete.

The screen shows the deletion and prompts you for the next expandable symbol:

```

with text_io;
use text_io;
procedure hello is
    <<<declarative_part>>>
begin
    <<sequence_of_statements>>
    <<<exception_handler_section>>>

```

```
end hello;
```

<<<declarative_part>>>: E(xpand), D(elete), B(ackup) or Q(uit)

16. Select Delete.

The screen shows the deletion and displays a choice menu for the next expandable symbol:

```
with text_io;
use text_io;
procedure hello is
begin
  <<statement>>
    <<sequence_of_statements>>
  <<<exception_handler_section>>>
end hello;
  <<statement>>
    <<simple_statement>>
    <<labeled_simple_statement>>
  BACKUP TO PREVIOUS SYMBOL
```

17. Select <<simple_statement>>.

The screen shows the expansion and displays a choice menu for the next expandable symbol:

```
with text_io;
use text_io;
procedure hello is
begin
  <<simple_statement>>
    <<<sequence_of_statements>>>
  <<<exception_handler_section>>>
end hello;
  <<simple_statement>>
  <<assignment_statement>>
    <<if_statement>>
    <<procedure_call_statement>>
    <<while_statement>>
    <<for_statement>>
    <<loop_statement>>
    <<return_statement>>
    <<case_statement>>
    <<block_statement>>
    <<raise_statement>>
    <<exit_statement>>
    <<null_statement>>
    <<delay_statement>>
    <<accept_statement>>
    <<entry_call_statement>>
    <<select_statement>>
    <<abort_statement>>
```

```

<<code_statement>>
<<goto_statement>>
BACKUP TO PREVIOUS SYMBOL

```

18. Select <<procedure_call_statement>>.

The screen shows the expansion and displays a choice menu for the next expandable symbol:

```

with text_io;
use text_io;
procedure hello is
begin
    <<real_name>><<actual_parameter_part>>;
    <<sequence_of_statements>>
<<exception_handler_section>>
end hello;

    <<real_name>>
    <<simple_name>>
    <<real_name>>.<<simple_name>>
BACKUP TO PREVIOUS SYMBOL

```

19. Select <<simple_name>>.

The screen shows the expansion and displays a choice menu for the next expandable symbol:

```

with text_io;
use text_io;
procedure hello is
begin
    <<identifier>><<actual_parameter_part>>;
    <<sequence_of_statements>>
<<exception_handler_section>>
end hello;

    <<identifier>>
    <<INPUT_identifier?>>
    <<letter>><<more_identifier>>
Examples
TBD
a_more_readable_name
BACKUP TO PREVIOUS SYMBOL

```

20. Select <<INPUT_identifier>>.

You are prompted for an identifier:

Input expression:

21. Type put_line and press ENTER.

The screen shows your substitution and prompts you for the next optional expansion:

```

with text_io;
use text_io;
procedure hello is
begin

```

```

    put_line<<actual_parameter_part>>;
    <<sequence_of_statements>>
<<exception_handler_section>>
end hello;

```

<<actual_parameter_part>>: E(xpand), D(etele), B(ackup) or Q(uit).

22. Select Expand.

The screen shows the expansion and prompts you for the next option:

```

with text_io;
use text_io;
procedure hello is
begin
    put_line(<<formal_parameter>> => ><<actual_parameter>><,
<<parameter_association>>>);
    <<sequence_of_statements>>
<<exception_handler_section>>
end hello;

```

<<formal_parameter>> => >: E(xpand), D(etele), B(ackup) or Q(uit)

23. Select Delete.

The screen shows the deletion and displays a choice menu for the next expandable symbol:

```

with text_io;
use text_io;
procedure hello is
begin
    put_line(<<actual_parameter>><, <<parameter_association>>>);
    <<sequence_of_statements>>
<<exception_handler_section>>
end hello;
<<actual_parameter>>
    <<expression>>
    <<variable_name>>
    <<type_mark>>(<<variable_name>>)
    BACKUP TO PREVIOUS SYMBOL

```

24. Select <<expression>>.

The screen shows the expansion and displays a choice menu for the next expandable symbol:

```

with text_io;
use text_io;
procedure hello is
begin
    put_line(<<expression>><, <<parameter_association>>>);
    <<sequence_of_statements>>

```

```

<<exception_handler_section>>
end hello;
    <<expression>>
    <<INPUT_expression?>>
    <<relation>>
    <<relation>> and <<and_expression>>
    <<relation>> and then <<and_then_expression>>
    <<relation>> or <<or_expression>>
    <<relation>> or else <<or_else_expression>>
    <<relation>> xor <<xor_expression>>
    BACKUP TO PREVIOUS SYMBOL

```

25. Select <<INPUT_expression>>.

You are prompted for an expression:

Input expression:

26. Type "hello" (include the quotation marks) and press ENTER.

The screen shows your substitution and prompts you for the next optional expansion:

```

with text_io;
use text_io;
procedure hello is
begin
    put_line("hello"<, <<parameter_association>>);
    <<sequence_of_statements>>
<<exception_handler_section>>
end hello;

```

<, <<parameter_association>>: E(xpand), D(etele), B(ackup) or Q(uit)

27. Select Delete.

The screen shows the deletion and prompts you for the next option:

```

with text_io;
use text_io;
procedure hello is
begin
    put_line("hello");
    <<sequence_of_statements>>
<<exception_handler_section>>
end hello;

```

<<sequence_of_statements>>: E(xpand), D(etele), B(ackup) or Q(uit)

28. Select Delete.

The screen shows the deletion and prompts you for the next optional expansion:

```

with text_io;
use text_io;

```

```

procedure hello is
begin
  put_line("hello");
<<<exception_handler_section>>>
end hello;

```

```

<<<exception_handler_section>>>: E(xpand), D(ete), B(ackup) or
Q(uit)

```

29. Select Delete.

The screen now shows the completed Ada program (**hello.ada**):

```

with text_io;
use text_io;
procedure hello is
begin
  put_line("hello");
end hello;

```

In the next tutorial you can compile, link, and run the Ada program.

3.7 Compiling, Linking, and Running an Ada Program

In the last tutorial, you used the ACE Language Symbol Expansion Feature to write an Ada program. In this tutorial, other ACE functions are used to compile, link, and run the program.

If you didn't go through that tutorial, use the ACE editor to type in this program, exactly as shown:

```

with text_io;
use text_io;
procedure hello is
begin
  put_line("hello");
end hello;

```

3.7.1 Compiling an Ada Program

Before you can compile a file from a given directory, you need to create an Ada Program library in that directory. You can do that with the **Library => Create New Library** function.

In a moment the program library is ready. Read the file **hello.ada** into an Edit Window. Then continue with step 1 below.

1. At the Edit Window containing your Ada program (**hello.ada**), start compilation.

Compile=>Compile File

A series of commands appears at the bottom of the screen. In a short time, the program is compiled. The last message should be:

```

No compilation errors
Press any key to continue

```

If any errors were found, you see an Execute and Review Window combination that displays the program and an explanation of the error. Use that explanation and compare your file with the version shown above to find the specific problem. Then correct the error in the Execute Window and, from there, repeat step 1.

2. Press any key.

You are returned to editing mode in the same window.

3.7.2 Linking an Ada Program

After successful compilation, the next step is to link the program.

1. At the Edit Window containing your compiled Ada program, select the function to link the program.

Compile=>Link Program

The DOS command that links the program appears at the bottom of the screen, and in a moment the link is complete. If there is any problem during linking, the error window appears containing the error message. For example:

Program too big to fit in memory.
Press any key to continue

If this happens, press a key to continue. Resolve the problem and repeat Step 1 above.

3.7.3 Running an Ada Program

The payoff, after successfully writing, compiling, and linking a program, is to run it. That's what you do in this tutorial.

1. From any Edit Window, select the function to run a program.

Run=>Run Program

A prompt appears asking for you to identify the program:

Program:

2. Enter the program name.

hello <ENTER>

Immediately the program is executed and the results display just under the name you entered:

hello (Running...)

hello

Press Enter to Continue

Press ENTER to return to the Edit Window from which you ran the program.

3.8 Debugging an Ada Program

In the last two tutorials you created, compiled, linked, and ran a simple Ada program.

In this tutorial, you introduce an error into the same program and try to compile it. The resulting sequence shows you how to use ACE functions to identify, examine, and correct program compilation errors. Finally, you verify the correction by recompiling the program successfully.

Start with the Ada program file, `hello.ada`, created in an earlier tutorial. It looked like this:

```
with text_io;
use text_io;
procedure hello is
begin
  put_line("hello");
end hello;
```

Open an Edit Window with this file in it. (If you didn't complete the earlier tutorials, type it in, exactly as shown.) Then delete the quotation marks around "hello" (the argument to `put_line`) in line 5. When you do so, line 5 should be:

```
put_line(hello);
```

Take the steps below to attempt to compile this program, and then to fix it using ACE Execute and Review Windows:

1. At the Edit Window containing your Ada program (`hello.ada`), start compilation.

Compile=>Compile File

The Ada compiler command appears at the bottom of the screen. In a short time, the error is detected and this split-screen display appears:

Execute Window

```
with text_io;
use text_io;
procedure hello is
begin
  _ put_line(hello);
end hello;
```

Review Window

```
Meridian Ada Compiler [v4.1 Aug. 4, 1990] Target 80X86
"HELLO.ADA", 5: type clash in parameter to "put_line" [LRM 6.4/1]
7 lines compiled.
1 error detected.
```

The Execute and Review Window combination displays the program and an explanation of the error.

In the Execute Window, the cursor is flashing at the beginning of the line containing the error (i.e., `..put_line(hello);`). In the Review Window, the highlighted line identifies the type of error and cites the pertinent LRM section (i.e., LRM 6.4/1).

2. Correct the error in the Execute Window.

<Move the cursor to (hello) and change it to ("hello")>

When the file is corrected, proceed with step 4.

3. Close the Execute & Review Window to return to your original source file.

<Shift-F2>

Menu: Window=>Close Window

4. Recompile the program.

Compile => Compile File

In a few moments the hello program (hello.ada) is compiled and the following message appears at the bottom of the screen:

No compilation errors

Press any key to continue

Your debugging was successful. If you didn't complete the earlier tutorials on linking and running a program, you can turn back to them and do that now.

That ends the ACE tutorials. We hope you found them helpful, and that you feel confident about using the system. If you need any other information about the system, you should be in a position to find it by referring to Chapter 4 or the online Help.

Chapter 4 ACE Features

4.1 Introduction

All ACE features are introduced in this chapter. These descriptions include all the editor and compiler system interface functions and their associated options. This chapter is organized in the following manner:

- Each menu item located on the Main Menu has its own section.
- The documentation for each submenu is documented under its Main Menu item.
- Menus are listed in the order they appear on the Main Menu.

In the step by step instructions, the menu instructions are given in the text and the function keys are shown in parentheses immediately following.

4.2 Editor Functions

ACE editor functions enable users to perform a range of operations including cursor movement, key binding, text processing and formatting, graphic illustration, file and buffer management, window access and control, operating system level activities, and online help.

4.3 Ada Compiler System Functions

ACE also includes functions for Ada program development, compilation, and execution, Ada Program library access, and program linking operations.

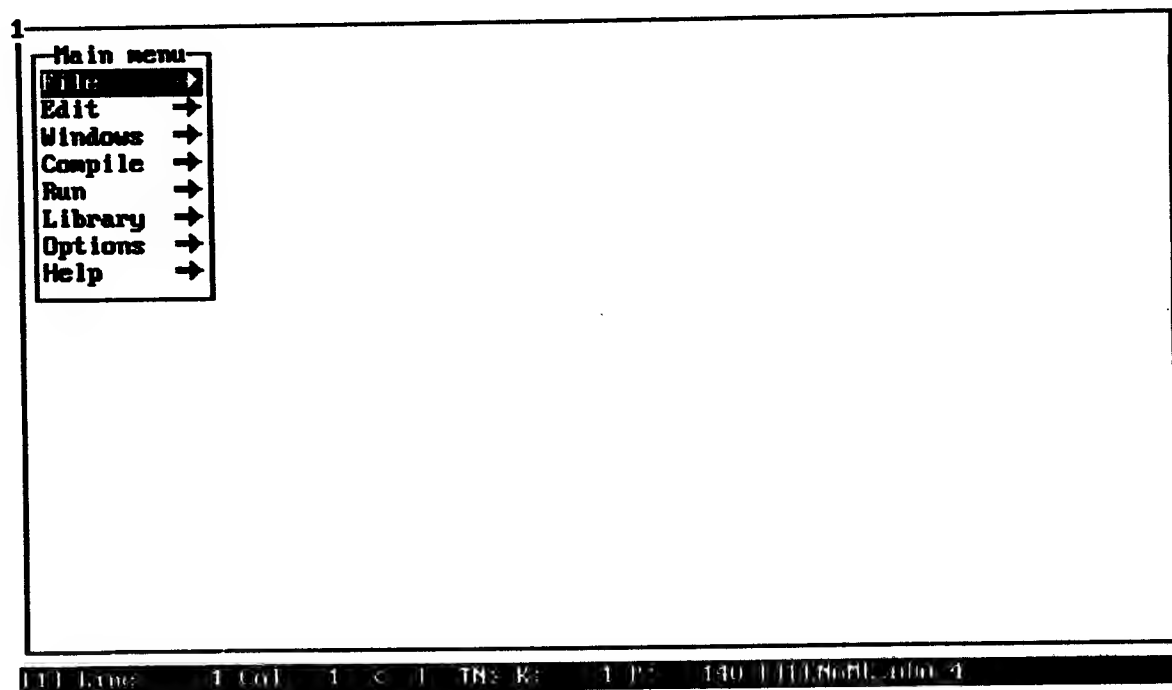
ACE is an environment built around the Meridian Ada Compiler. Previous releases of the compiler did not have an environment; the compiler was invoked from the DOS prompt (command line).

The compiler documentation is based on the concept of commands being issued from the command line.

As you read the ACE documentation you will notice references such as "For more information, see the *xxxxx - x* command in the *Meridian Ada Compiler User's Guide*." In ACE the compiler functions are no longer broken down by command, but by function and are accessed by menus. That is, a command with or without an option in previous releases of the compiler without ACE, can now be performed with ACE using the menus.

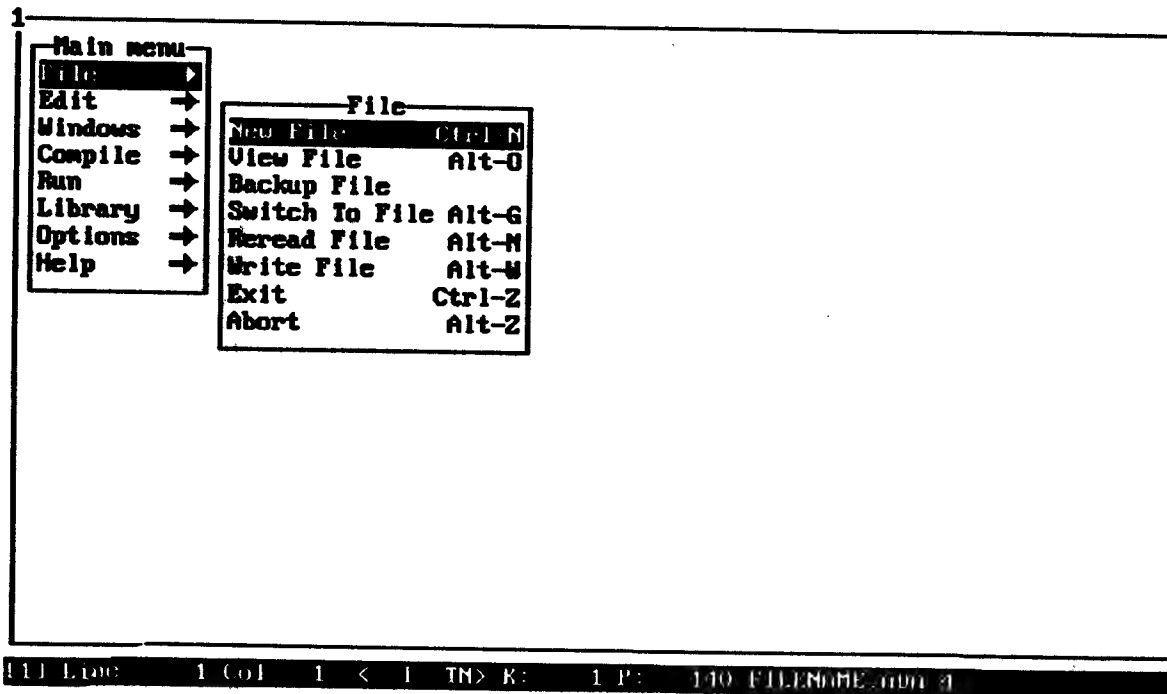
The *Meridian Ada Compiler User's Guide* contains more detailed information about each command/function. Where possible each compiler function documented in this manual has a reference to a command which is documented in detail in the *Meridian Ada Compiler User's Guide*. Please refer to the *Meridian Ada Compiler User's Guide* for detailed information on the compiler functions. You can also use the online help feature to view additional information about compiler related functions.

4.4 Main Menu



The Main menu provides access to the File, Edit, Windows, Compile, Run, Library, Options, and Help menus.

4.5 File Menu



This menu provides access to a range of file manipulation functions that support ACE operations. Some of the file functions available from this menu allow you to create a new file, view a file, write a file, exit ACE, and abort a session.

Menu Selections

New File (Ctrl-N)

This function allows you to select a file to edit in the current window. This can be an existing file or a completely new file.

To create a new file, follow these steps:

1. Access the File menu, select New File, and press ENTER. You are prompted:

Filename:

2. Type the filename and press ENTER. You are prompted again:

File does not exist. Create?
Yes
No

3. Highlight Yes (or type Y) and press ENTER. The file is created and displayed.

ACE Features

View File (Alt-O)

This function opens a file for read-only viewing. To view a file, follow these steps:

1. Access the File Menu, select View File, and press ENTER. You are prompted:

Filename:

To see a list of existing files, press ENTER.

2. Type or select the filename and press ENTER. The file is displayed.

Backup File

Backup File copies the current file to a file with the same name in a subdirectory called **ACEBACK**. This file can be used to abort editing without saving your file changes. Also, if you must reboot your computer while editing, this file is a backup. The default is for ACE to automatically keep a backup file.

You can specify that a backup is made automatically using the Backup function. You can also specify whether the backup file is to be deleted or retained at the end of each editing session using the function Retain Backup Across Sessions.

To backup a file, access the File Menu, select Backup File, and press ENTER. The current file is backed up and the current screen reappears.

Switch to File (Alt-G)

This function selects the text at the current cursor location and switches you to that file for editing.

Reread File (Alt-N)

This function accesses the Restore File screen used to reread the current file into the window.

1. Access the File Menu, select Reread File and press ENTER. You are prompted:

Restore File path\filename:
Yes
No

Write File (Alt-W)

This function writes the text associated with the current window to a file. To write a file, follow these steps:

1. Access the File menu, select Write File, and press ENTER. You are prompted:

Filename:

2. Type the filename and press ENTER. The text is written to the file and you are returned to the previous screen.

Exit (Ctrl-Z)

This function exits from ACE, saving the current file changes.

Abort (Alt-Z)

This function restores the current file and exits from ACE.

1. When you select Abort or use Alt-Z, ACE displays the following prompts:

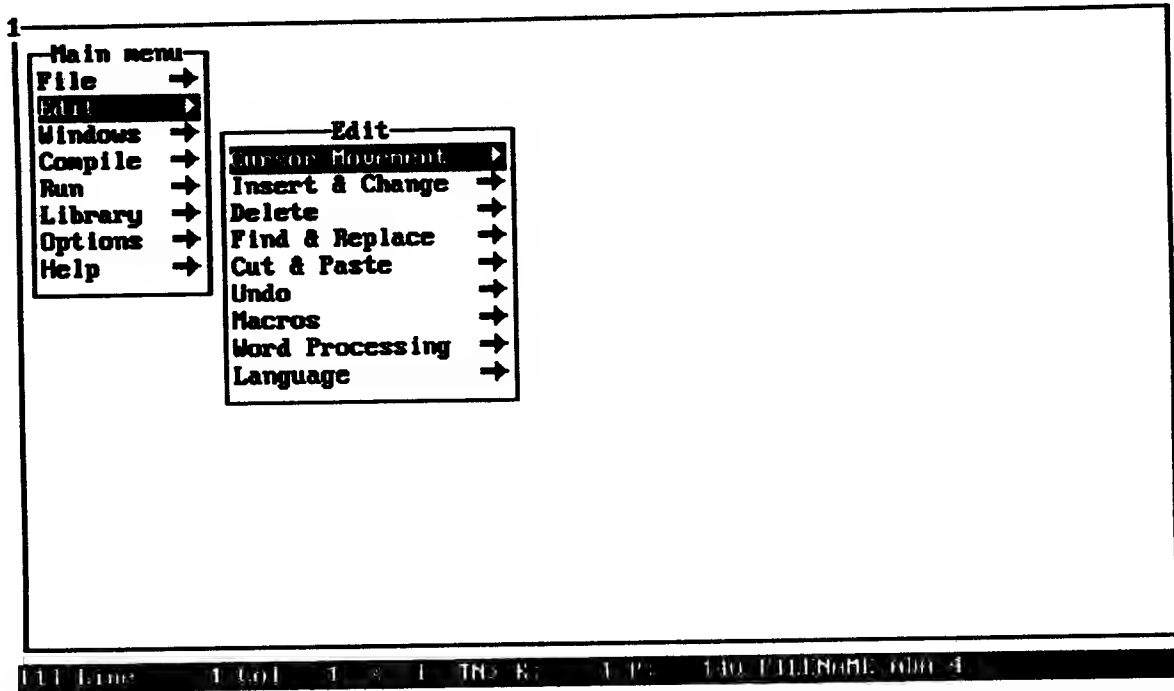
Abort Editing?
Yes
No

2. If you answer Yes to this prompt and have made changes to any file during the current editing session and a backup file has been created, ACE displays the following prompt:

Restore *directory\filename*

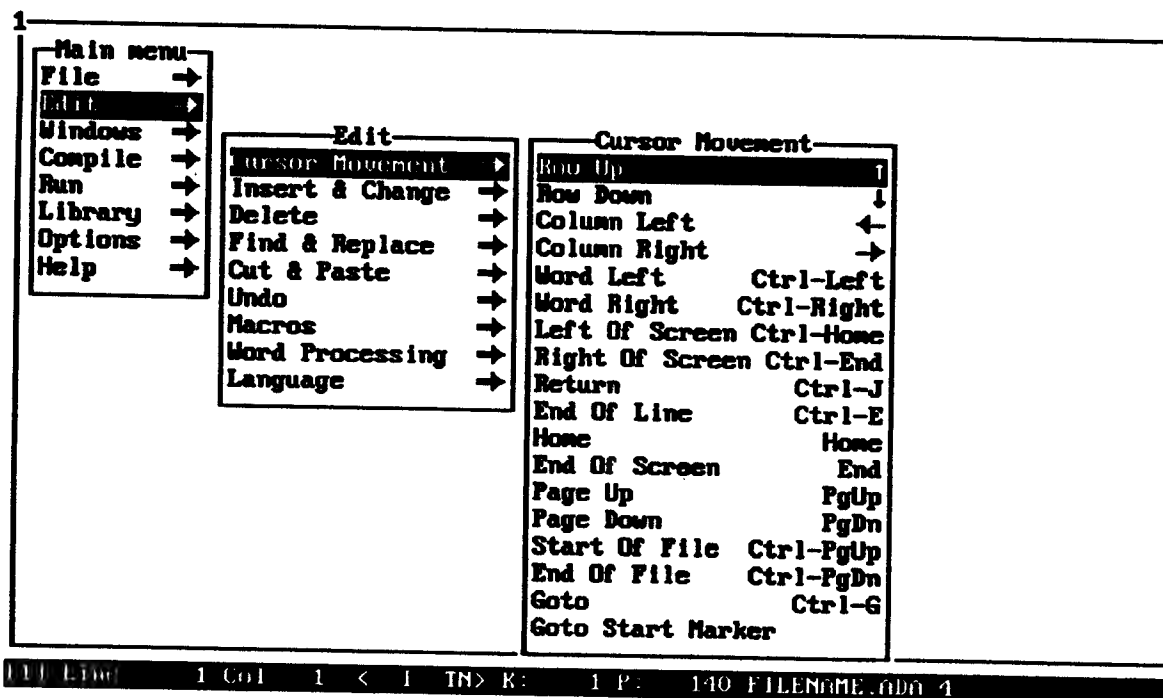
Yes, restores the file to its last saved state. No exits without restoring it; whatever changes you have made are preserved.

4.6 Edit Menu



The Edit Menu allows you to access menus which allow you to perform cursor movement, insert and change functions, find and replace functions, and cut & paste functions. You can also use this menu to create macros and perform word processing.

4.6.1 Cursor Movement



The Cursor Movement Menu allows you to move the cursor up/down a page, to the start/end of the file, to a specific line, to the left/right of the screen, and to a premarked location.

To move the cursor to a particular file location, access the Cursor Movement Menu, select a function (e.g., Page Up), and press ENTER. The cursor is moved to the new location (e.g., up one page). The screen shows the cursor in the new location.

Menu Selections

Row Up (up arrow ↑)

This function moves the cursor up one line in the window. If the cursor is at the top of the window and you are not at the top of the file, the window moves up one line.

Row Down (down arrow ↓)

This function moves the cursor down one line in the window. If the cursor is at the bottom of the window, the window moves down one line.

Column Left (left arrow ←)

This function moves the cursor left one column. When the cursor reaches the window edge and you are not at the right margin of the file, the window shifts.

Column Right (right arrow →)

This function moves the cursor right one column. When the cursor reaches the window edge, the window shifts.

Word Left (Ctrl- ←)

This function moves the cursor left one word. If the next word is not in the window, the window is repositioned with the word centered in the window.

Word Right (Ctrl- →)

This function moves the cursor right one word. If the next word is not in the window, the window is repositioned with the word centered in the window.

Left of Screen (Ctrl-Home)

This function moves the cursor to the left edge of the window.

Right of Screen (Ctrl-End)

This function moves the cursor to the right edge of the screen.

Return (Ctrl-J)

This function shifts the window to the left edge and places the cursor in the first column and down a line.

End of Line (Ctrl-E)

This function moves the cursor to the end of the current line.

Home (Home)

This function moves the cursor to the upper left corner of the window.

End of Screen (End)

This function moves the cursor to the bottom of the window.

Page Up (PgUp)

This function moves the text up the length of the window.

Page Down (PgDn)

This function moves the text down the length of the window.

Start of File (Ctrl-PgUp)

This function moves the cursor to the beginning of the file.

End of File (Ctrl-PgDn)

This function moves the cursor to the end of the file.

Goto (Ctrl-G)

This function moves the cursor to a specified line in the file. To move the cursor to a specific location in the file, follow these steps:

1. Access the Cursor Movement Menu, select Goto, and press ENTER. You are prompted:

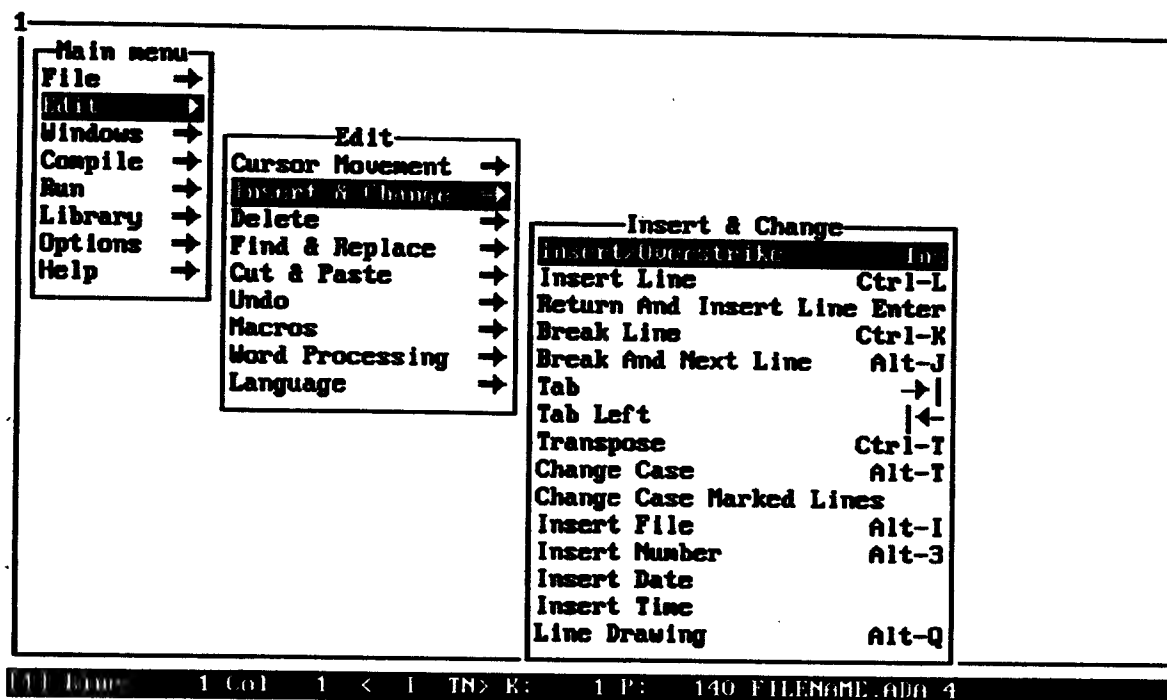
Line Number:

2. Type the line number and press ENTER. The cursor is moved to the specified line number and the screen display changes, showing the new cursor location.

Goto Start Marker

This function moves the cursor to the start of a previously marked area of text. To move the cursor to the start of a marked area (defined by the Mark Lines or Mark Columns functions), access the Cursor Movement Menu, select Goto Start Marker, and press ENTER. The cursor is moved to the beginning of the previously marked area. See section 4.6.5 for information about marking lines or columns.

4.6.2 Insert & Change Menu



The Insert & Change Menu provides access to those editing operations which insert new text (or open space) into a file or modify existing text. Because most of these procedures are self-explanatory, the following pages only include directions for a few of the more complex operations.

Menu Selections

Insert/Overstrike (Ins)

This function toggles between the insert and overstrike character modes. In the overstrike mode, back-space does not drag text with it. In the line drawing mode, this function is ignored and overstriking is always performed. An I located in the status line between the brackets < > indicates insert mode is on. The absence of an I indicates overstrike mode is turned on.

Insert Line (Ctrl-L)

This function inserts a blank line at the current line.

Return And Insert Line (ENTER)

This function moves the cursor to the next line and inserts a blank line there.

Break Line (Ctrl-K)

This function breaks the line into two lines at the cursor position.

Break and Next Line (Alt-J)

This function breaks the line at the cursor position into two lines and moves the cursor to the next line.

Tab (tab key)

This function moves the cursor to the next tab stop. If Insert Tabs is on, then spaces are inserted to the next tab stop. Insert Tabs is set on the Edit Options Menu.

Tab Left (shift-tab key)

This function moves the cursor left to the previous tab stop.

ACE Features

Transpose (Ctrl-T)

This function swaps the current character with the next character.

Change Case (Alt-T)

This function changes the current character's case according to the Change Case Mode. To set the Change Case Mode, see section 4.11.2.

Change Case Marked Lines

This function changes the marked lines of text according to the Change Case Mode. To change the case (upper <=> lower) one or more lines, follow these steps:

1. Using the arrow keys, move the cursor to the beginning of the text to be changed.
2. Access the Cut & Paste Menu, select Mark Lines, and press ENTER.

Marking is turned on and the menus disappear.

3. Using the arrow keys, move the cursor over the area to be case changed. The area under the cursor is marked (highlighted) on the screen.
4. Access the Insert and Change Menu, select Change Case Marked Lines, and press ENTER.

The menu disappears and the marked lines are changed. Letters are changed according to the Change Case Mode option on the Settings Menu. For more information, see section 4.11.2.

Insert File (Alt-I)

This function inserts a file into the current file at the cursor line.

Insert Number (Alt-3)

This function inserts a previously established number (set through Insert # on the Settings Menu) into the file. The insert number is always incremented by one each time it is used.

Insert Date

This function inserts the current date into the file at the cursor position.

Insert Time

This function inserts the current time into the file at the cursor position.

Line Drawing (Alt-Q)

This function turns on the line drawing mode, allowing you to use cursor keys to construct line drawings. To create a line drawing, follow these steps:

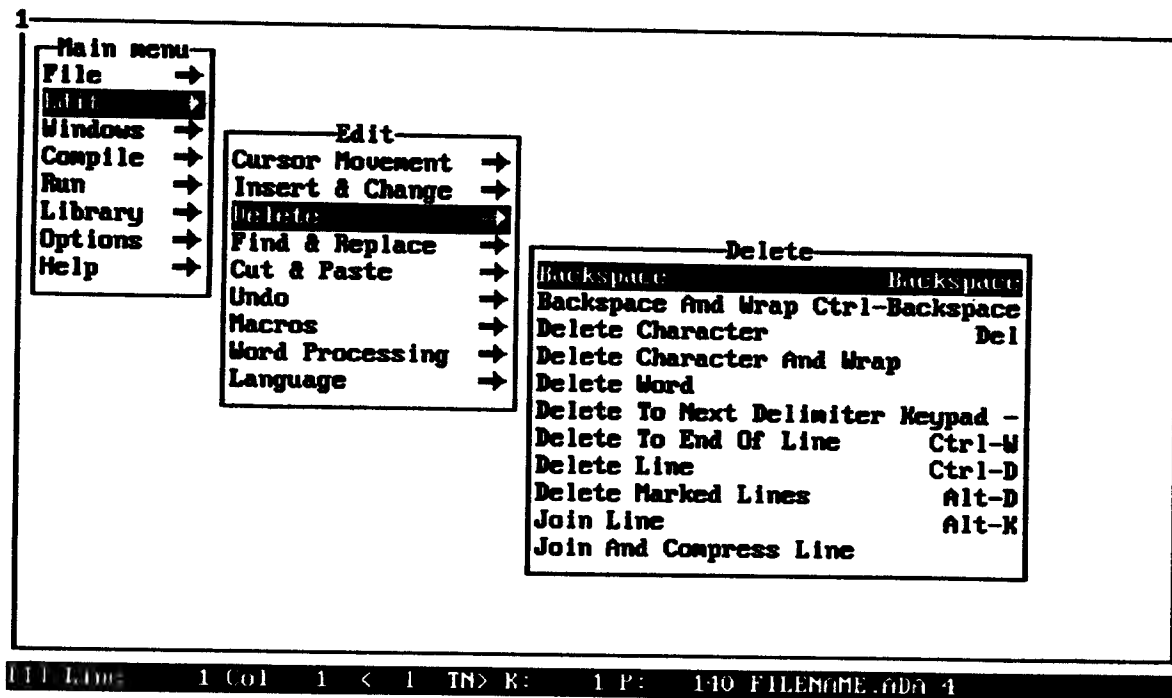
1. Access the Insert & Change Menu, select Line Drawing, and press ENTER.

The Line Drawing mode is now active on the current screen.

2. Using Alt-Q to turn on/off the drawing mode (while you move the cursor to a new position) and the arrow keys (up/down, left/right) to move the tracing cursor, construct the line drawing. When the drawing is complete, press ENTER.

While the drawing mode is active, the arrow keys trace a white line on the screen. After ENTER is pressed, the drawing mode is turned off.

4.6.3 Delete Menu



The Delete Menu provides access to those editing operations that generally delete some text from the current file. Because most of these procedures are self-explanatory, one step actions, the following pages only include directions for a few of the more complex operations.

Menu Selections

Backspace (Backspace key)

This function deletes the previous character. If the cursor is at column one, nothing is deleted.

Backspace and Wrap (Ctrl-Backspace)

This function backspaces the cursor, deleting previous characters. If the cursor is at the beginning of a line, the current line is joined to the previous line.

Delete Character (Del)

This function deletes the current character.

Delete Character and Wrap

This function deletes the current character. If the cursor is at the end of a line, the next line is joined to the current line.

Delete Word

This function deletes the next word forward.

Delete to Next Delimiter (keypad -)

This function deletes to the next delimiter. You may set several delimiter characters with the Set Delimiters function of the Options Menu. To delete text to the next delimiter, follow these steps:

1. Using the arrow keys, move the cursor to the beginning of the area to be deleted.
2. Access the Edit Menu, select Delete to Next Delimiter, and press ENTER.

Text is deleted up to the next delimiter.

Delete to End of Line (Ctrl-W)

This function deletes text from the cursor position to the end of the current line.

Delete Line (Ctrl-D)

This function deletes the current line, copying it to the kill buffer. The kill buffer can hold only about 25 lines depending on the lengths of the lines.

Delete Marked Lines (Alt-D)

This function deletes the marked lines.

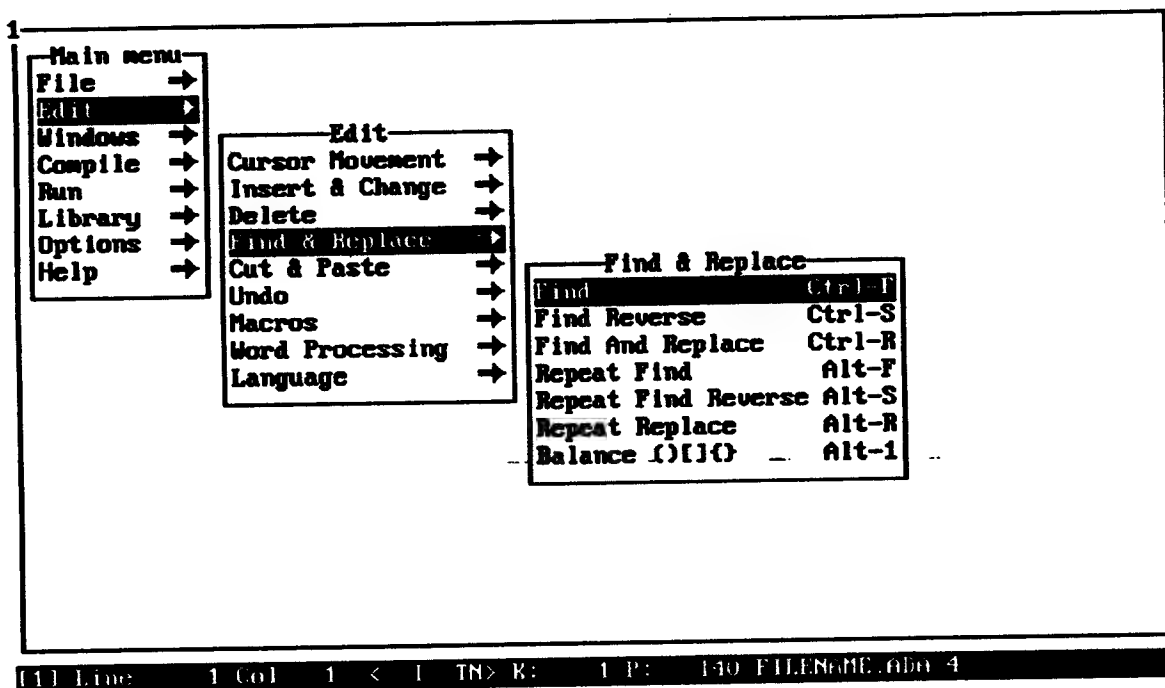
Join Line (Alt-K)

This function joins the next line to the current line, leaving a space between the lines.

Join and Compress Line

This function joins the next line to the current line, without leaving a space between the lines.

4.6.4 Find & Replace Menu



The Find & Replace Menu is used to search the file for a string and optionally replace it with another string.

The complete procedure for Finding and Replacing character strings follow:

To find a character string, follow these steps:

1. Access the Find & Replace Menu, select Find, and press ENTER. You are prompted:

Find:

2. Type the character string you are trying to find and press ENTER. The cursor moves to the line containing the specified string.

If the string is not located, the following message appears:

```
<string> not found
Press a key to continue
```

In this case, press any key to return to the previous screen.

To replace a character string, follow these steps:

1. Access the Find & Replace Menu, select Find and Replace, and press ENTER. You are prompted:

```
Replace:
```

2. Type the character string to be replaced and press ENTER. You are prompted:

```
Replace: <string> with:
```

3. Type the replacement string and press ENTER. ACE shows you the selected string for replacement along with the replacement string and asks if you want to change just this occurrence ((Y)es), don't replace ((N)o), replace all occurrences ((A)ll), or stop replacing ((Q)uit).

If the original string is not located, the following message appears:

```
<string> not found
Press a key to continue
```

In this case, press any key to return to the previous screen.

Menu Selections

Find (Ctrl-F)

This function is used to search the file for a string.

Find Reverse (Ctrl-S)

This function searches for a string in the file from the current cursor position back to the beginning of the file.

Find and Replace (Ctrl-R)

This function is used to replace a string.

Repeat Find (Alt-F)

This function repeats the last Find operation.

Repeat Find Reverse (Alt-S)

This function repeats the last Find Reverse operation.

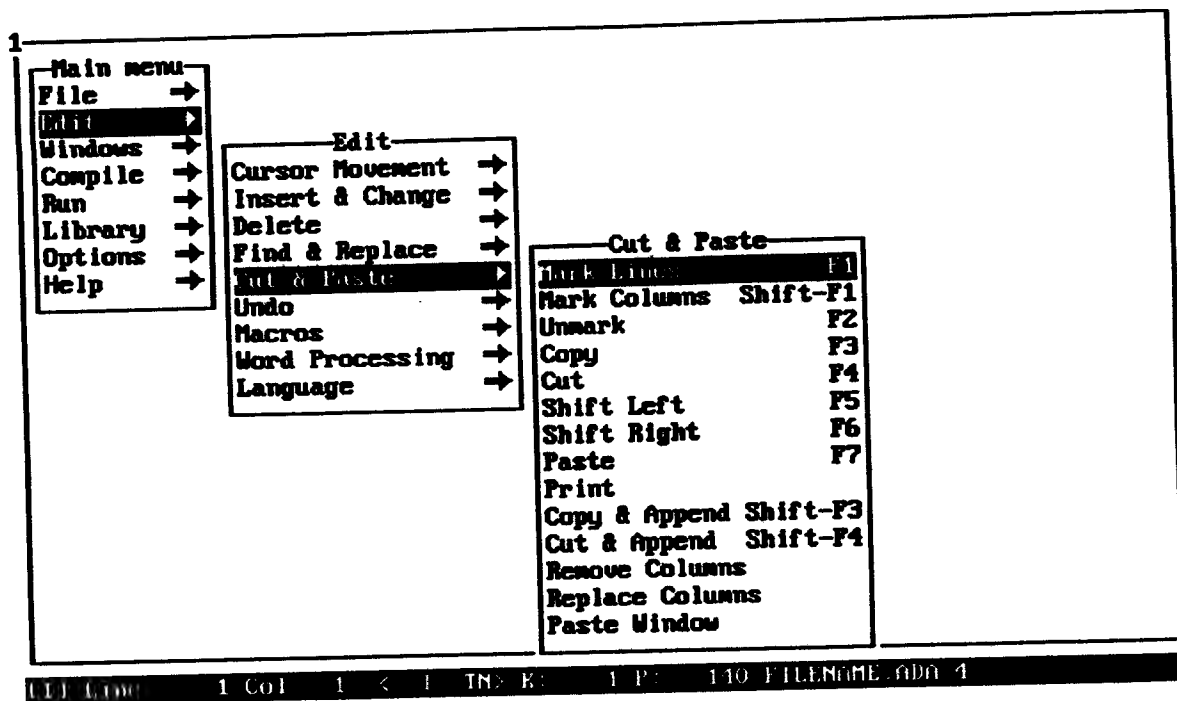
Repeat Replace (Alt-R)

This function repeats the last Replace operation.

Balance (Alt-1)

This function is used in an attempt to balance `{[(or)]}` with the corresponding `}` or `[{`.

4.6.5 Cut & Paste Menu



The Cut & Paste Menu is used for various cut & paste operations. You can mark and copy or delete lines/columns, save cut material (in the paste buffer), and print marked areas of text.

Note: The way that the marked text was defined determines how the text is subsequently pasted into the file. Also, the way that the text was cut or copied determines whether the paste buffer was cleared or not. When the text was defined by Mark Lines, the text is pasted in a typical line oriented arrangement. When the text was defined by Mark Columns, the text is pasted in a column arrangement. Copy or Cut operations clear the paste buffer. On the other hand, Copy & Append or Cut & Append do not clear the paste buffer.

Menu Selections

Mark Lines (F1)

This function marks one or more lines of text for Cut & Paste operations. With the cursor located at the beginning or end of the text you wish to mark, access the Mark Lines function on the menu or press F1. Then use the up or down arrow keys to select the lines of text.

Mark Columns (Shift-F1)

This function marks one or more columns of text for Cut & Paste operations. With the cursor located at the beginning or end of the text you wish to mark, access the Mark Columns function on the menu or press Shift-F1. Use right arrow key to define the horizontal boundary of the column and the down arrow key to select the lines.

Unmark (F2)

This function resets the current marker (canceling the previous mark).

Copy (F3)

This function copies the contents of the marked text into the paste buffer, deleting any previous text from the buffer. To copy text from a file to the paste buffer and clear the previous contents of the paste buffer, follow these steps:

1. Using the arrow keys, move the cursor to the beginning of the text to be copied.
2. Access the Cut & Paste Menu, select Mark Lines or Mark Columns, and press ENTER.
The Cut & Paste Menu disappears, leaving the screen showing the text file.
3. Using the arrow keys, move the cursor over the text to be marked.
The text under the cursor is marked (highlighted) on the screen.
4. Access the Cut & Paste Menu again, select Copy, and press ENTER.
The Cut & Paste Menu disappears while the marked text is copied to the paste buffer. Simultaneously, the buffer is cleared of all previously included text.

Now, you can access the Cut & Paste Menu again to paste the copied text to another place in the current file or just save the copied text (in the buffer) for later pasting to the same file or a different file. See Paste which is described later in this section.

Cut (F4)

This function deletes the marked text and puts it into the paste buffer, deleting previous text from the buffer. To cut text from a file, copy it to the paste buffer, and clear the buffer of all previously included text, follow these steps:

1. Using the arrow keys, move the cursor to the beginning of the text to be cut.
2. Access the Cut & Paste Menu, select Mark Lines or Mark Columns, and press ENTER.
The Cut & Paste Menu disappears, leaving the screen showing the text file.
3. Using the arrow keys, move the cursor over the text to be marked.
The text under the cursor is marked (highlighted) on the screen.
4. Access the Cut & Paste Menu again, select Cut, and press ENTER.
The Cut & Paste Menu disappears while the marked text is cut from the file and copied to the paste buffer. Simultaneously, the buffer is cleared of all previously included text.

Now, you can paste the cut text to another place in the current file or save the cut text (in the buffer) for later pasting to the same file or a different file. See Paste which is described later in this section.

Shift Left (F5)

This function shifts the marked text (or the current line if no text is marked) to the next tab stop on the left. To shift a column to the right or left in a file, follow these steps:

1. Using the arrow keys, move the cursor to the upper left of the column to be moved.
2. Access the Cut & Paste Menu, select Mark Columns, and press ENTER. The Cut & Paste Menu disappears, leaving the screen showing the file.
3. Using the arrow keys, move the cursor over the column (from the upper left to the lower right) to be moved. The column under the cursor is marked (highlighted) on the screen.
4. Access the Cut & Paste Menu again, select Shift Left (or Shift Right), and press ENTER. The marked column is moved to the left (or right) in the file while the Cut & Paste Menu disappears from the screen.

Shift Right (F6)

This function shifts the marked text (or the current line if no text is marked) to the next tab stop on the right. See Shift Left for more information.

Paste (F7)

This function copies the content of the paste buffer into the current file at the cursor position. To paste previously cut or copied text into a file, follow these steps:

1. Using the arrow keys, move the cursor to the place where the text is to be pasted.
2. Access the Cut & Paste Menu, select Paste, and press ENTER. The Cut & Paste Menu disappears and the contents of the paste buffer (previously cut or copied text) is pasted to the file at the cursor position.

Print

This function prints the marked text. To print the contents of the marked text, follow these steps:

1. Using the arrow keys, move the cursor to the beginning of the text to be cut.
2. Access the Cut & Paste Menu, select Mark Lines, and press ENTER.

The Cut & Paste Menu disappears, leaving the screen showing the text file.

3. Using the arrow keys, move the cursor over the text to be marked. The text under the cursor is marked (highlighted) on the screen.
4. Access the Cut & Paste Menu again, select Print, and press ENTER. The contents of the marked text are printed.

Copy & Append (Shift-F3)

This function copies the contents of the marked text into the paste buffer without deleting any previous text from the buffer. To copy text from a file to the paste buffer without clearing the buffer, follow these steps:

1. Using the arrow keys, move the cursor to the beginning of the text to be copied.
2. Access the Cut & Paste Menu, select Mark Lines, and press ENTER.

The Cut & Paste Menu disappears, leaving the screen showing the text file.

3. Using the arrow keys, move the cursor over the text to be marked.

The text under the cursor is marked (highlighted) on the screen.

4. Access the Cut & Paste Menu again, select Copy & Append, and press ENTER.

The Cut & Paste Menu disappears while the marked text is copied to the paste buffer. The newly copied text is appended to text previously included in the buffer.

Now, you can access the Cut & Paste Menu again to paste the contents of the paste buffer (including the newly copied text) to another place in the current file or just save it for later pasting to the same file or a different file.

Cut & Append (Shift-F4)

This function deletes the marked text and puts it into the paste buffer without deleting any previous text from the buffer. To cut text from a file and copy it to the paste buffer without clearing the buffer, follow these steps:

1. Using the arrow keys, move the cursor to the beginning of the text to be cut.
2. Access the Cut & Paste Menu, select Mark Lines, and press ENTER.

The Cut & Paste Menu disappears, leaving the screen showing the text file.

3. Using the arrow keys, move the cursor over the text to be marked.

The text under the cursor is marked (highlighted) on the screen.

4. Access the Cut & Paste Menu again, select Cut & Append, and press ENTER.

The Cut & Paste Menu disappears while the marked text is cut from the file and copied to the paste buffer. The newly cut text is appended to text previously included in the buffer.

Now, you can access the Cut & Paste Menu again to paste the contents of the paste buffer (including the newly cut text) to another place in the current file or just save it for later pasting to the same file or a different file.

Remove Columns

This function copies the marked columns to the paste buffer and clears the columns in the current file. That is, the text is removed and the column(s) are filled with blanks, maintaining the original format. To remove column(s) from a file and copy them to the paste buffer, follow these steps:

1. Using the arrow keys, position the cursor at the upper left of the column(s) to be removed.
2. Access the Cut & Paste Menu, select Mark Columns, and press ENTER.

The Cut & Paste Menu disappears, leaving the screen showing the text file.

3. Using the arrow keys, move the cursor over the column(s) (from the upper left to the lower right) to be marked. The column under the cursor is marked (highlighted) on the screen.
4. Access the Cut & Paste Menu again, select Remove Columns, and press ENTER. The Cut & Paste Menu disappears while the marked column(s) are removed from the file and copied to the paste buffer.

Now, you can access the Cut & Paste Menu again to replace other columns, in the current file, with the newly removed column(s) (now in the paste buffer) in the currently displayed file or just save it for later pasting to the same file or a different file.

Replace Columns

This function copies the column(s) in the paste buffer into the current file, overlaying the previous text. To replace column(s) with previously removed column(s), follow these steps:

1. Using the arrow keys, move the cursor to the location where the new column(s) are to be placed.
2. Access the Cut & Paste Menu, select Replace Columns, and press ENTER.

The Cut & Paste Menu disappears and the contents of the paste buffer (the previously removed column(s)) are pasted to the file at the cursor position.

Paste Window

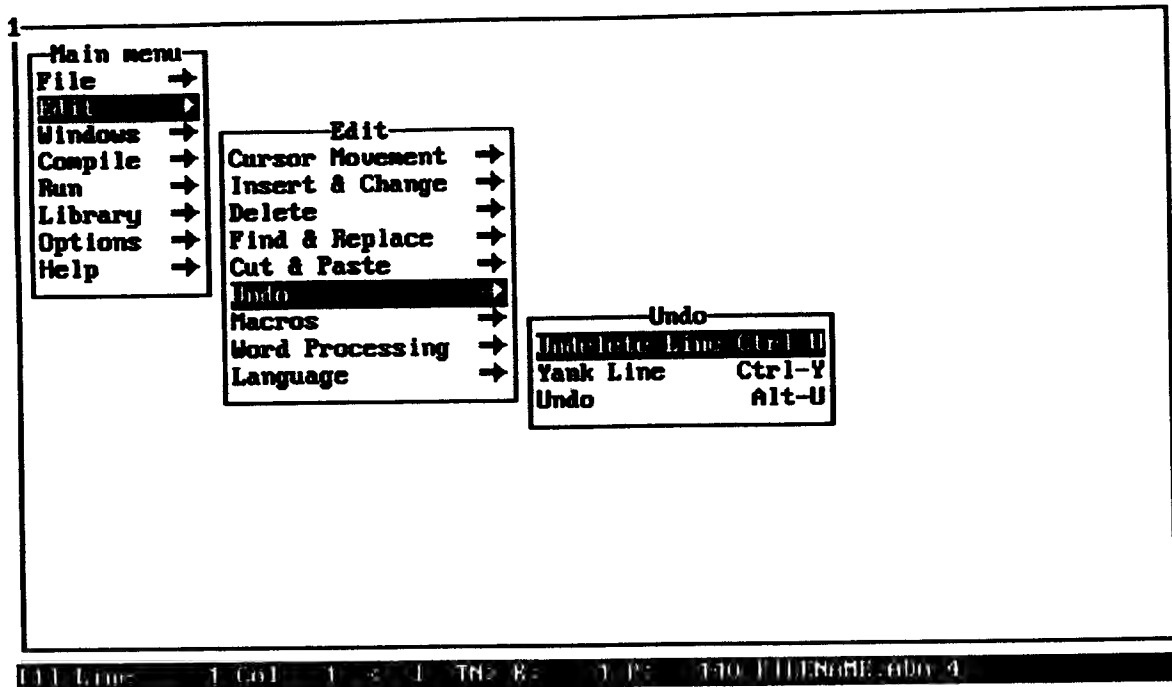
This function accesses the Paste Window for editing. To access the Paste Window, follow these steps:

1. Access the Cut & Paste Menu, select Paste Window, and press ENTER.

The Paste Window appears.

Now, you can edit the paste buffer through the window, adding, deleting, and changing the contents before the buffer is pasted to a file or printed.

4.6.6 Undo Menu



The Undo Menu provides access to functions which allow you to recover previously deleted or otherwise modified text.

Menu Selections

Undelete Line (Ctrl-U)

This function inserts a line from the kill buffer at the current line. That line is deleted from the kill buffer.

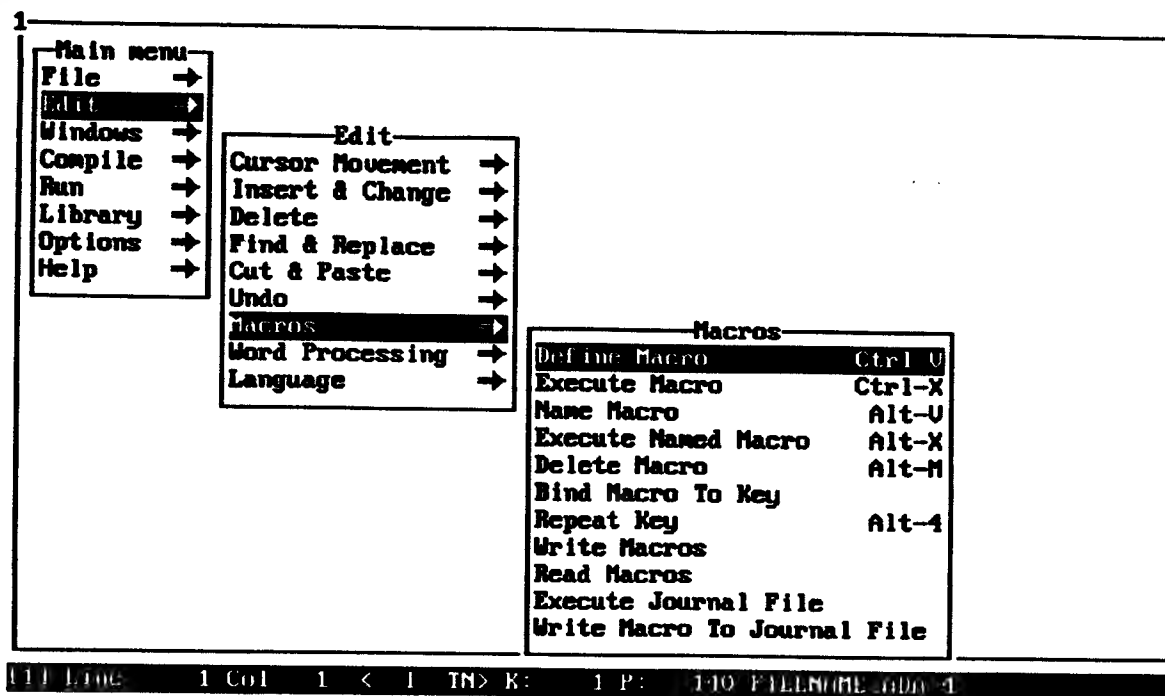
Yank Line (Ctrl-Y)

This function inserts a line from the kill buffer at the current line. The line is not deleted from the kill buffer, allowing it to be yanked multiple times.

Undo (Ctrl-U)

This function reverses the changes you have made to the current file. PgUp and PgDn, will undo all changes made during an editing session. If logging is off, then undo only restores the current line to the last edited state.

4.6.7 Macros Menu



This menu is used to set and change keystroke macros and manipulate the Journal File.

Menu Selections

Define Macro (Ctrl-V)

This function starts and stops keystroke recording. To define a macro, follow these steps:

1. Access the Macros Menu, select Define Macro, and press ENTER (or simply, press Ctrl-V).
The system is set to record keystrokes and you are returned to the previous screen
2. Type the key stroke sequence, then select Define Macro (or press Ctrl-V) again to turn off recording. An example might be selecting a character string and replacing it with a new character string. The sequence (between Define Macro selections) is recorded.

These macros can be named and saved in a macro file to be recalled later using the "Execute Named Macro" function described below.

Execute Macro (Ctrl-X)

This function executes the current keystroke macro.

Name Macro (Alt-V)

This function assigns a name to the current keystroke macro. To define a macro, see "Define Macro" above.

Execute Named Macro (Alt-X)

This function executes a named macro. To execute a previously created and named macro, follow these steps:

1. Access the Macros Menu, select Read Macros, and press ENTER. You are prompted:

Read macro file [default=ACE.MAC]:

2. Type the macro filename (e.g., **abc.mac**), and press ENTER. The macro file is read and a message appears:

Reading macro file

3. Access the Macro Menu and select Execute Named Macro (or simply press Alt-X). You are prompted:

Select Macro

<macro name>

4. Leave the macro name highlighted and press ENTER.

The macro is executed in the current window at the cursor position.

Delete Macro (Alt-M)

This function deletes a named macro.

Bind Macro to Key

This function binds a macro to a key.

Repeat Key (Alt-4)

This function repeats a key the number of times indicated.

Write Macros

This function writes the named macros to a file. If no filename is given, the macro is written to the default file name. See Execute Named Macro for more information.

Read Macros

This function reads in a macro file. If no filename is given, the default filename is used. This file contains your macros.

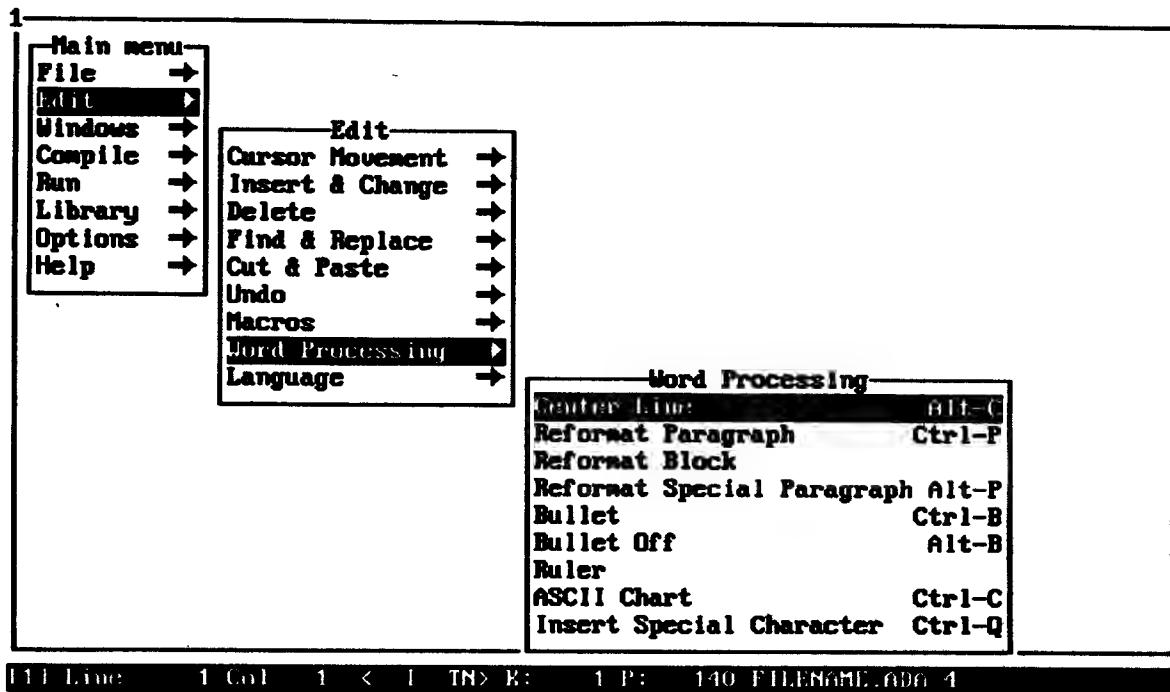
Execute Journal File

This function executes a journal file.

Write Macro to Journal File

This function writes a macro, in ASCII form, to a journal file.

4.6.8 Word Processing Menu



The Word Processing Menu is used to access a range of text handling functions (e.g., paragraph reformatting).

Menu Selections

Center Line (Alt-C)

This function centers the current line, using the right margin.

Reformat Paragraph (Ctrl-P)

This function reformats text from the beginning of the current line to the next empty line or to the end of the file. The current right margin and indentation of the next line are used. With this function, the first line can be differently indented than the rest of the paragraph. To reformat a paragraph, follow these steps:

1. Using the arrow keys, move the cursor to the first line of the text to be reformatted.
2. Access the Word Processing Menu, select Reformat Paragraph, and press ENTER.

The paragraph is reformatted.

Reformat Block

This function reformats text from the beginning of the current line to the next empty line (or to the end of the file) using the current right margin and the indentation of the current line.

Reformat Special Paragraph (Alt-P)

This function reformats text the same way the Reformat Paragraph does, but without changing the number of spaces between the first and second word. This function is used for bulleted lists.

Bullet (Ctrl-B)

This function sets the left margin for word wrapping, overriding the auto-tab setting. Also, see "Auto-Tab".

Bullet Off (Alt-B)

This function resets the left margin to the first column.

Ruler

This function, displaying a ruler at the bottom of the screen, is used to set soft tabs and the right margin.

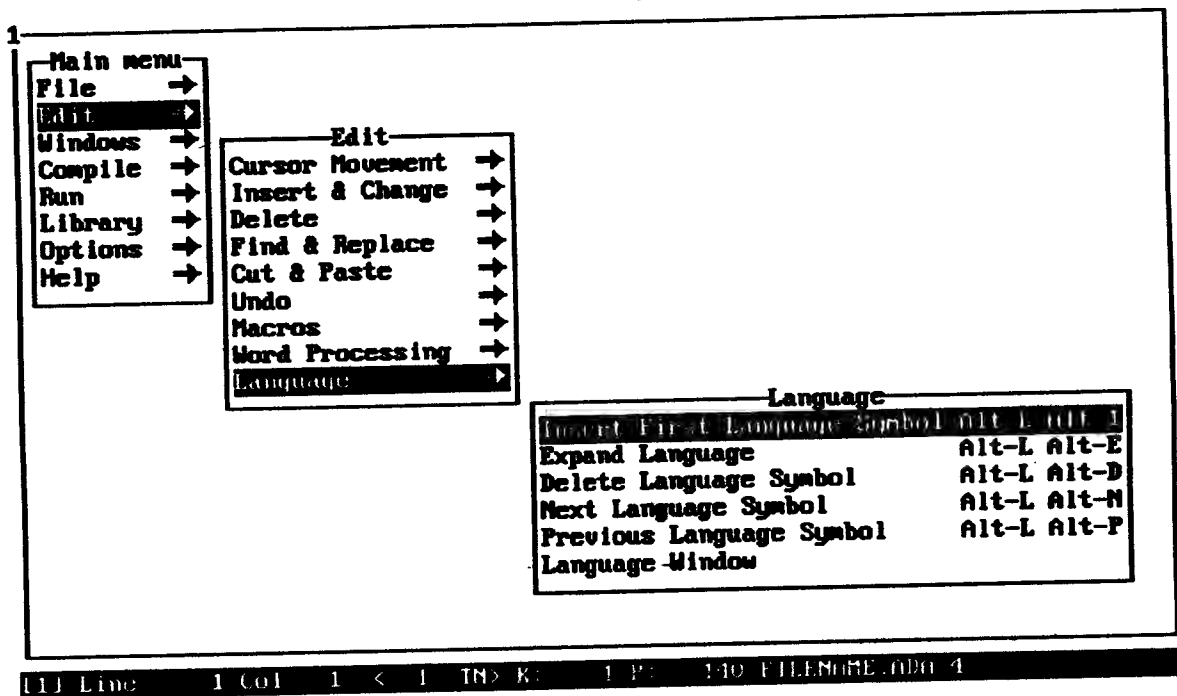
ASCII Chart (Ctrl-C)

This function, expresses ASCII values in decimal, hexadecimal, and character format.

Insert Special Character (Ctrl-Q)

This function inserts any ASCII character specified by a decimal number.

4.6.9 Language Menu



The Language Menu assists you in developing Ada programs. The Insert First Language Symbol, Expand Language, Delete Language Symbol, Next Language Symbol, and Previous Language Symbol can be used to develop Ada programs from scratch or expand existing programs. These procedures are explained in section 3.6.

Menu Selections

Insert First Language Symbol (Alt-L Alt-1)

This function inserts the first language symbol into the language file.

Expand Language (Alt-L Alt-E)

This function looks up the language symbol at the current cursor position in the Language Window and expands it appropriately.

Delete Language Symbol (Alt-L Alt-D)

This function deletes the language symbol at the current cursor position.

Next Language Symbol (Alt-L Alt-N)

This function moves the cursor to the next language symbol.

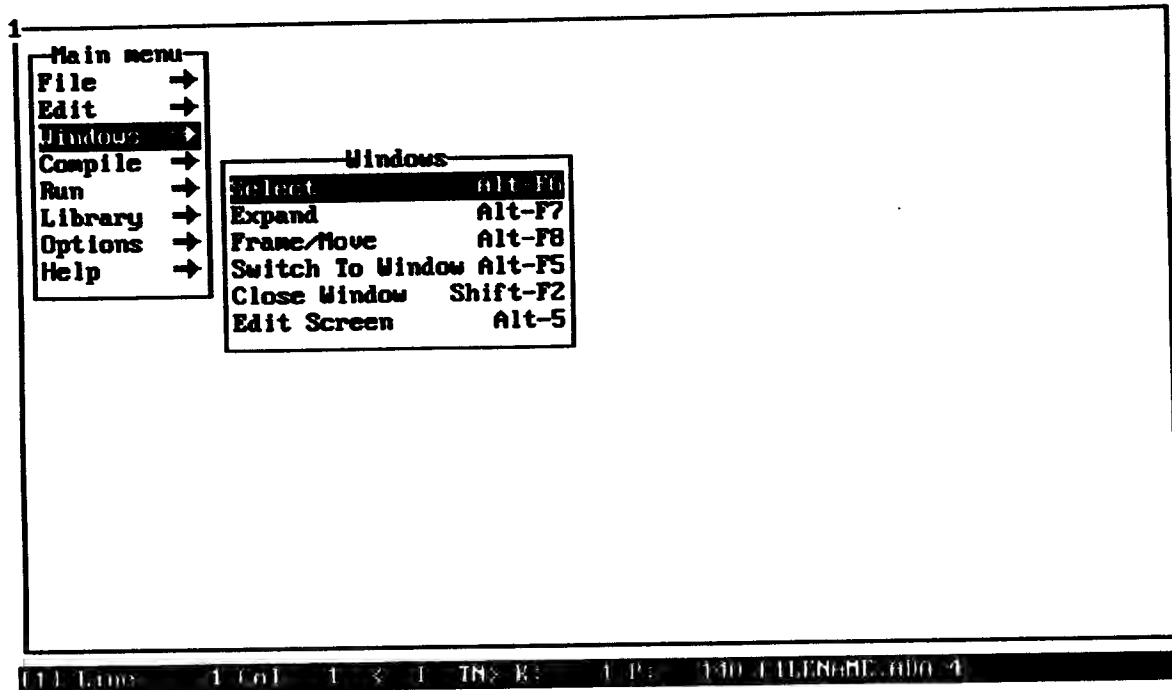
Previous Language Symbol (Alt-L Alt-P)

This function moves the cursor to the previous language symbol.

Language Window

This function accesses the Language Window used to define the language expansion rules.

4.7 Windows Menu



The Windows Menu is used to access specific windows and set window characteristics (e.g. window size).

Menu Selections

Select (Alt-F6)

This function accesses the Select Window screen. The Select Window screen is used to select windows (editing and specialized windows) for ACE operations.

To select a window, follow these steps:

1. Access the Select Menu, select a window (e.g., Execute Window), and press ENTER.

The selected window appears.

A list of windows follows.

#1 – 10

These items are used to access the ten general purpose editing windows. The currently accessed window name is blinking.

DOS Window

This item accesses the DOS Window used for DOS commands. For more information, see section 2.3.

Execute Window

This item accesses the Execute Window used to review compilation errors. For more information, see section 2.3.

Review Window

This item accesses the Review Window used to examine editing work and display error messages. For more information, see section 2.3.

Paste Window

This item accesses the Paste Window used to temporarily store text for copy/paste operations. For more information, see section 2.3.

Language Window

This item accesses the Language Window used in Ada language expansion. For more information, see section 2.3.

Kill Window

This item accesses the Kill Window used to store deleted text. For more information, see section 2.3.

Expand (Alt-F7)

This function expands the current window to full screen size.

Frame/Move (Alt-F8)

This function is used to change the current window size and location. This function has two modes, Frame and Move. Frame allows you to resize the window and Move allows you to move the window. To toggle between the two modes, use the ENTER key. To reset the size and location of the current window, follow these steps:

1. Access the Windows Menu, select Frame/Move, and press ENTER.

The Windows Menu disappears while the current window reappears surrounded by a rectangular frame.

Following the directions provided below (and at the bottom of the screen), you may resize the window frame or move the window to a different place on the screen.

To resize the screen frame:

1. Using the arrow keys, make the frame horizontally or vertically smaller/larger (moving the cursor left/right or up/down, respectively). Press the appropriate arrow keys, then press ESC.

The window resizes as defined by you.

To split the screen:

1. Using the Control Key (Ctrl) with other keys, split the screen frame into a rectangle on the left/right or at the top/bottom of the screen (Ctrl- →/Ctrl- ← or Ctrl-PgUp/Ctrl-PgDn, respectively).

Press the appropriate Control-key combination, then press ESC to set the new window arrangement.

The screen splits with the smaller rectangle positioned at the left, right, top, or bottom of the screen, depending upon the Control-key command.

Switch to Window (Alt-F5)

This function is used to switch to another window. To switch to another window, follow these steps:

1. Access the Windows Menu, select Switch to Window, and press ENTER. You are prompted:

Window Number:

2. Type the window number and press ENTER. You are prompted again:

Filename:

To see a list of existing files, press ENTER.

3. Type or select a filename and press ENTER.

The Windows Menu disappears while the specified window appears.

Close Window (Shift-F2)

This function closes the currently displayed window. If other windows are still open (in the background), the last window used before the current window is displayed. To close the current window, follow these steps:

1. Access the Windows Menu, select Close Window, and press ENTER.

The Windows Menu disappears and the current window closes, returning you to the last window opened (before the current window).

However, if you are at the first window opened in the current editing session, you are prompted:

[window] Filename:

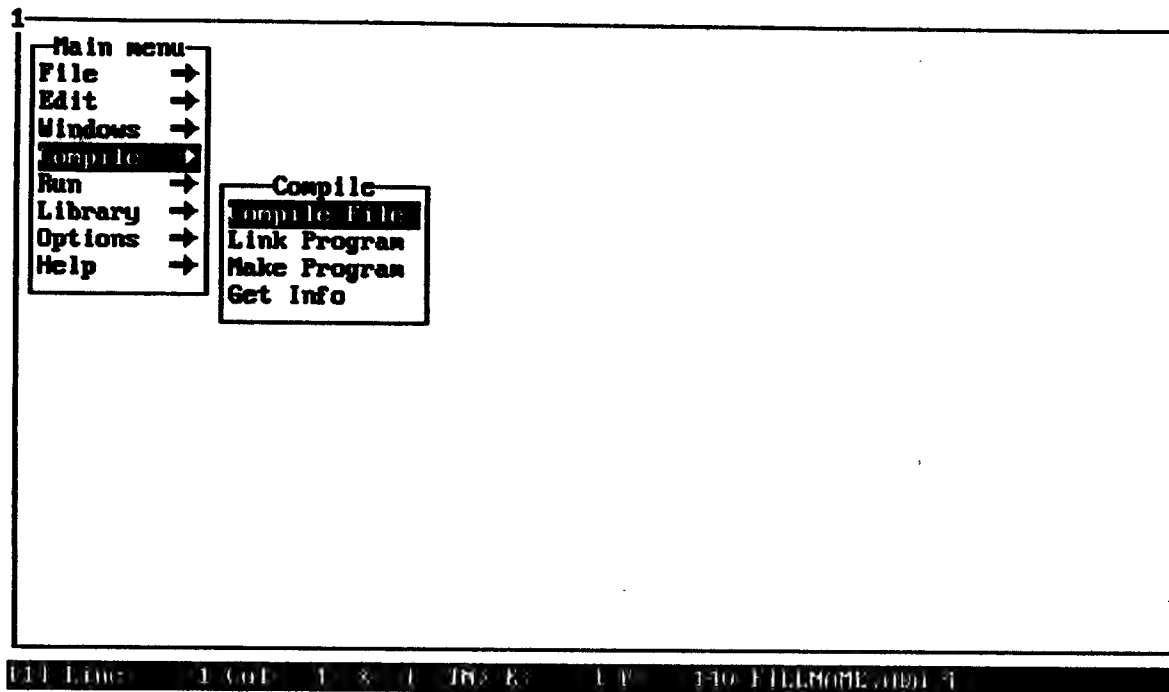
In this case, type a filename or press ENTER.

If you entered a filename, the corresponding window appears. If you just pressed ENTER, a list of all filenames appears, allowing you to select a file.

Edit Screen (Alt-5)

This function allows you to review and edit the screen that was present when ACE started.

4.8 Compile Menu



The Compile Menu is used to compile a file (Ada program), link a program, remake a program, or obtain information about the compilation. The command line equivalents for each of these functions are described in detail in the *Meridian Ada Compiler User's Guide*. Please refer to that manual for more detailed information on how and when to use these functions.

Menu Selections

Compile File

This function is used to compile an Ada source file. This function performs the equivalent action of the **ada** command. To compile a file (Ada program), follow these steps:

1. Access the Compile Menu, select Compile File, and press ENTER.

The file in the current window (e.g., **sample.ada**) is compiled.

If there are errors in your Ada program, the screen displays the Execute Window above the Review Window. The Review Window shows the first error highlighted, for example:

```
sample.ada, 2: undefined package in use clause "direct_io" (LRM 10.1.1)
```

In the Execute Window, the cursor is at the beginning of the line containing the error.

If you have errors, you can correct them in the displayed Execute Window. To recompile the program, return to your original source file window with the Close Window function (Shift-F2) and reissue the Compile File function.

See section 3.7 for more introductory information about compilation and the *Meridian Ada Compiler User's Guide* for a complete explanation of this process and the **ada** command.

Link Program

This function is used to link an Ada program. This function performs the equivalent action of the **bamp** command. To link the Ada program, follow these steps:

1. Access the Compiler Menu, select Link Program, and press ENTER.

The Ada program (e.g., **sample**) is linked. In the process, an executable program is created with the name of the current Ada main subprogram.

This procedure is generally described in section 3.7.

See the *Meridian Compiler User's Guide* for a complete explanation of linking and the **bamp** command.

Make Program

This function automatically recompiles compiles and relinks your program when the Ada program source code changes. This function performs the equivalent action of the **amake** command. To remake you program, follow these steps:

1. Access the Compiler Menu, select Make Program, and press ENTER.

The current Ada main subprogram (e.g., **sample**) is recompiled and relinked.

See the *Meridian Compiler User's Guide* for a complete explanation of linking and the **amake** command.

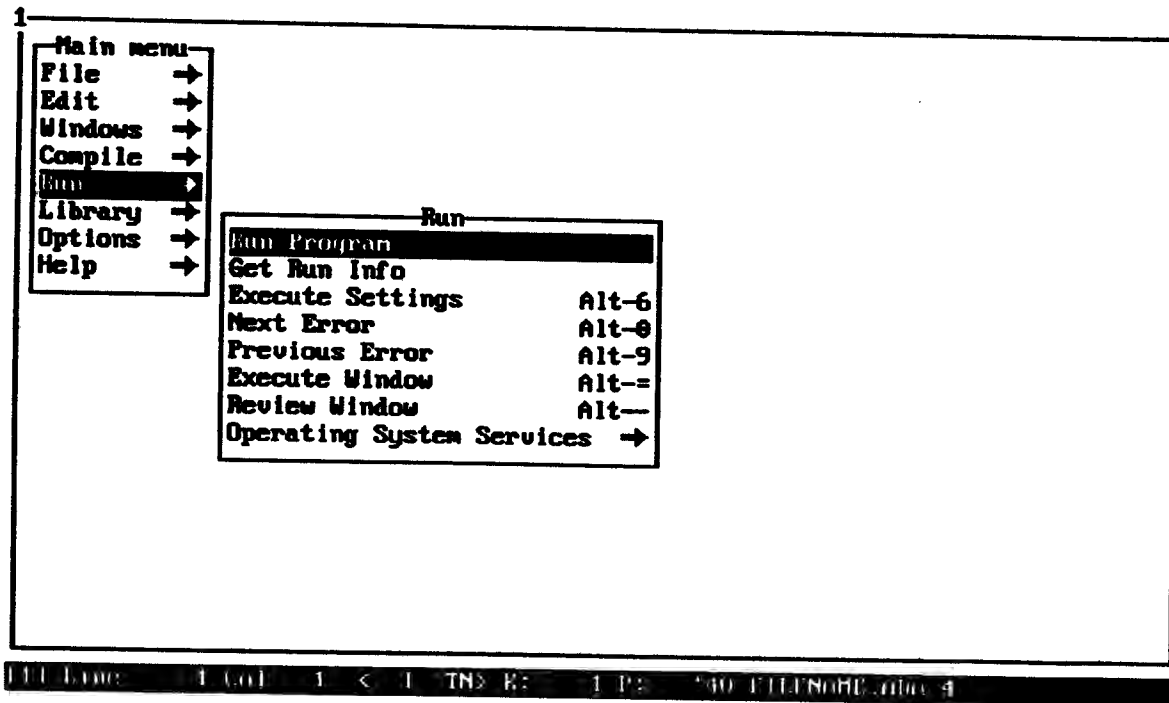
Get Info

This function obtains information about the current compilation. To obtain information about the results of the most recent compile, link, or make function, follow these steps:

1. Access the Compile Menu, select Get Info, and press ENTER.

The Review Window is displayed, containing the most recent results.

4.9 Run Menu



The Run Menu is used to execute programs and obtain information about program execution. This menu should be used after the Run Options Menu is used to set program execution options.

Menu Selections

Run Program

This function runs a program. To run (execute) a Program, follow these steps:

1. Access the Run Menu, select Run, and press ENTER. You are prompted:

Program:

2. Type the program name (e.g., `sample`) and press ENTER or if the file you want to run is displayed in the current window, press ENTER.

The program is run (executed). The program name is displayed at the bottom of the screen.

Refer to the *Meridian Ada Compiler User's Guide* for more information about the extended program execution command (`xamp`). Also, see "Run Options" in this manual.

Get Run Info

This function is used to obtain the results of the most recent invocation of the Run Program function. These results are only available if you turned the Save Standard Output option on when you ran the program. To obtain information about a program execution, follow these steps:

1. Access the Run Menu, select Get Run Info, and press ENTER.

The Review Window is displayed, containing the output of the most recent program execution.

Execute Settings (Alt-6)

This function allows you to create DOS command lines that can be submitted for execution from this screen. As you define each DOS command, it is added to the list and saved for future use. These commands can be as simple as a **DIR** command or as complicated as search and display type functions.

1. Access the Run Menu, select Execute Settings, and press ENTER.

A small window appears at the top of the screen with the following items:

Function This is the user defined function name. You can enter any name you want to give your executable string. (Optional field.)

Execute String This is the actual command line to be sent to DOS for execution. (Required field.)

Review File This identifies the filename you reference to review the results of executing the command line in Execute String. If it is set to (none) then any results will not be reviewed. This file must have been created by the execution of the Execute String. (Optional field.)

Output Window

This identifies the window you want to switch to and examine the Review file in when execution finishes. If it is set to (none) then the results if any will be reviewed in the current window. (Optional field.)

Swap Out This determines whether ACE should be swapped out while the command is executing. (Required field.)

Key Binding This identifies the key binding, if any, associated with this execute string. (Optional field.)

Now, you can edit these items, execute items, add items, delete items, or exit.

Note: The key stroke commands for these operations are provided at the bottom of the screen.

Defining/Editing an Item:

1. Highlight **Function**: and press ENTER. You are prompted for the function name. Enter the name you want to assign to this item.

Function:

2. Type in the requested information and press ENTER.

The window changes at the top of the screen, showing your new value for that field of the item.

3. When you have completed your definition, press ESC to exit or execute the item by pressing Ctrl-ENTER. In either case you will receive the following prompt:

Write Changed Settings To Config File?
Yes
No

4. If you accept the current setting, highlight Yes and press ENTER. If not, highlight No and press ENTER.

You are prompted:

Write config file [default=<path>\<config file>]:

5. If you accept the current setting, just press ENTER. If not, type in a different filename or path and press ENTER.

The setting is written to the configuration file and a message appears:

Writing config file

Adding an item:

1. Press Ctrl-L.

A new line appears with (none) in each field of the item. Define the new item using the "Defining/Editing an Item" description above.

Deleting an item:

1. Highlight any field in the item you wish to delete and press Ctrl-D.

The item and all its fields disappear.

Exiting:

1. At any point, press ESC. If you have made any changes, you receive the following prompt:

Write Changed Settings To Config File?
Yes
No

2. If you accept the current setting, highlight Yes and press ENTER. If not, highlight No and press ENTER. You are prompted:

Write config file [default=<path>\<config file>]:

3. If you accept the current setting, just press ENTER. If not, type in a different filename or path and press ENTER. The setting is written to the configuration file and a message appears:

Writing config file

The previous screen appears.

Next Error (Alt-0)

This function highlights the next error in the Review Window and positions the cursor at the corresponding line number in the appropriate file (displayed in the Execute Window).

To display the previous (or next) error, follow these steps:

1. Access the Run Menu, select Previous Error (or Next Error), and press ENTER.

The Review Window appears with the previous error (or next error since the last) displayed. If there was no error, the window is blank.

Previous Error (Alt-9)

This function highlights the previous error in the Review Window and positions the cursor at the corresponding line number in the appropriate file (displayed in the Execute Window).

Execute Window (Alt-=)

This function accesses the Execute Window used to examine the source code associated with error messages (simultaneously displayed in the Review Window, appearing beneath the Execute Window). For more information, see section 2.3.

To access the Execute (or Review) Window, follow these steps:

1. Access the Run Menu, select Review (or Execute) Window, and press ENTER.

You are prompted, for example:

[Review Window] Filename:

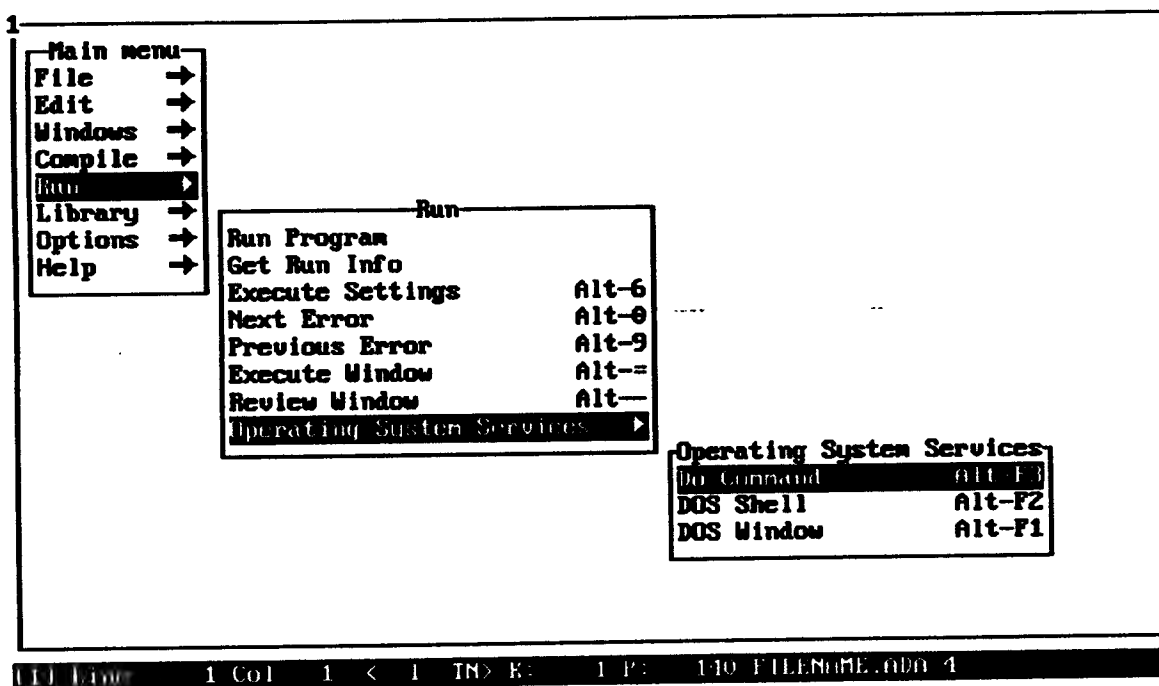
2. Type the window filename and press ENTER.

The Review (or Execute) Window appears with the file displayed. If you did not enter a filename, the Review (or Execute) Window appears blank.

Review Window (Alt=)

This function accesses the Review Window used to display compilation error messages. For more information, see "Using ACE Windows" in section 2.3. See "Execute Window" above for more information.

Operating System Services



This item accesses the Operating System Services Menu. The Operating System Services Menu provides access to operating system level features: a DOS window, a DOS subshell, and the ability to use the Do command.

Do Command (Alt-F3)

This function allows you to execute a single DOS command. To execute a DOS command from ACE, follow these steps:

1. Access the Operating System Services Menu, select Do Command, and press ENTER. The system prompts you:

DOS Command:

2. Type the DOS command (e.g., **chkdsk**) and press ENTER. The command is executed.

Executing: `command`

3. Press ENTER.

Your ACE Editing Window reappears.

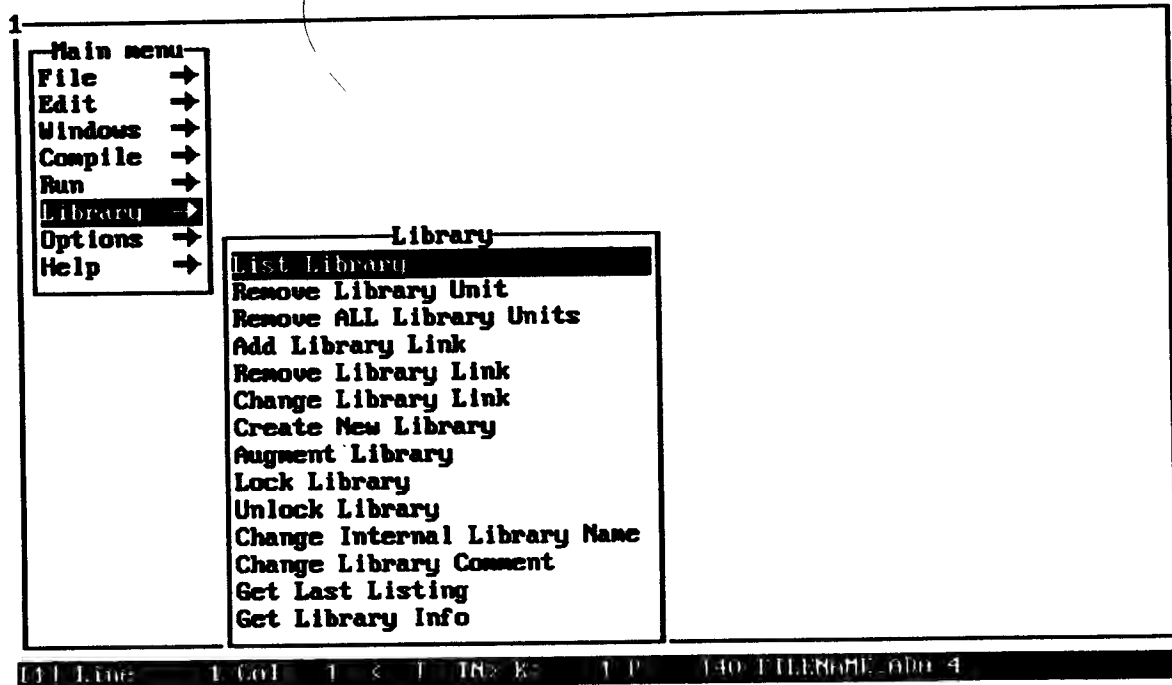
DOS Shell (Alt-F2)

This function swaps out the ACE Environment and Starts up a DOS shell (`command.com`). To return to ACE enter EXIT.

DOS Window (Alt-F1)

This function allows you to interact with DOS in a windowed environment.

4.10 Library Menu



The Library Menu is used to perform various Ada program library-related operations.

The command line equivalents for each of these functions are described in detail in the *Meridian Ada Compiler User's Guide*. Please refer to that manual for more detailed information on how and when to use these functions.

Menu Selections

List Library

This function produces a list of the units in the current program library. This function performs the equivalent action of the `lslib` command. To produce a library list, follow these steps:

1. Access the Library Menu, select List Library, and press ENTER. You are prompted:

```
List Unit(s):
```

2. Type the requested information (i.e., the unit-name(s) to list) and press ENTER.

The list is displayed in the Review Window, summarizing the library contents.

However, if an item you entered is not found, this message appears:

```
"unit name" is not in the library
```

Remove Library Unit

This function deletes the specified library unit(s) from the current program library. This function performs the equivalent action of the `rmlib` command. To remove an Ada program library unit, follow these steps:

1. Access the Library Menu, select Remove Library Unit, and press ENTER. You are prompted:

Remove Unit(s) :

2. Type the requested information (i.e., the compilation unit name) and press ENTER.
The unit named is removed from the specified program library. All files associated with the unit are deleted (e.g., object file) except for source files.

Remove All Library Units

This function deletes all the library units in the current program library after requesting user confirmation. This function performs the equivalent action of the `rm1ib -a` command.

Add Library Link

This function adds library references or links to a program library. This function performs the equivalent action of the `ln1ib` command. To add an Ada program library link, follow these steps:

1. Access the Library Menu, select Add Library Link, and press ENTER. You are prompted:

Add Library Link:

2. Type the requested information (i.e., library-file path name) and press ENTER.
The program library is linked. This will allow your Ada programs to reference compilation units defined in the library linked to.

However, if the item you entered is not found, this message appears:

cannot read program library "library-name"

Remove Library Link

This function deletes library links from a program library. This function performs the equivalent action of the `ln1ib -r` command. To remove an Ada program library link, follow these steps:

1. Access the Library Menu, select Remove Library Link, and press ENTER. You are prompted:

Remove Library Link:

2. Type the requested information (i.e., library-file path name) and press ENTER.
The link is deleted. (The associated library is not removed, only the link.)
However, if the link library-name you entered is not found, this message appears:
link not found: <library-name>

Change Library Link

This function deletes an existing library link from a program library and replaces it with a different link. This function performs the equivalent action of the `ln1ib -r` command. To change an existing Ada program library link, follow these steps:

1. Access the Library Menu, select Change Library Link, and press ENTER. You are prompted:

Old Library Link:

ACE Features

2. Type the requested information (i.e., the old library-file path name) and press ENTER. You are prompted :

New Library Link:

3. Type the requested information (i.e., the new library-file path name) and press ENTER.

The old library-name is replaced with the new library-name.

However, if the old item you entered is not found, this message appears:

link not found: <old library-name>

Create New Library

This function creates a new program library. This function performs the equivalent actions of the **mklib** and **lnlib** commands. To create a new Ada program library, follow these steps:

1. Access the Library Menu, select Create New Library, and press ENTER.

A new Ada program library is created.

However, if the compiler program executable file (**ada.exe**) cannot be found in the current execution search list specified by the **PATH** environment variable this message appears:

Cannot find compiler program ada.exe in search PATH variable

Augment Library

This function is used to attach additional link information to the library entry of a compilation unit. This function is primarily used in conjunction with pragma **interface**. This function performs the equivalent action of the **auglib** command. To attach additional link information to the library entry of an existing compilation unit, follow these steps:

1. Access the Library Menu, select Augment Library, and press ENTER. You are prompted:

Augment Unit:

2. Type the requested information (i.e., the existing library-unit) and press ENTER. You are prompted:

Link Parameters:

3. Type the requested information (i.e., the new link-parameters) and press ENTER.

However, if the library-unit you entered is not found, this message appears:

library unit "library-unit" not found

Lock Library

This function locks a program library. This function performs the equivalent action of the **modlib -l** command. This is a Groupware feature and is not available in all versions of the compiler. To lock a library, follow these steps:

1. Access the Library Menu, select Lock Library, and press ENTER.

The library is locked. While it is locked, the library cannot be updated by the compiler; however, it still can be used to link programs.

Unlock Library

This function unlocks a previously locked program library. This function performs the equivalent action of the **modlib -u** command. This is a Groupware feature and is not available in all versions of the compiler. To unlock a library that was previously locked, follow these steps:

1. Access the Library Menu, select Unlock Library, and press ENTER.

The library is locked.

Change Internal Library Name

This function is used to add text (up to 255 characters) to the program library header. This function performs the equivalent action of the **modlib -n** command. This is a Groupware feature and is not available in all versions of the compiler. To add text to a library header, follow these steps:

1. Access the Library Menu, select Change Internal Library Name, and press ENTER. You are prompted:

Library Name :

2. Type the additional text (up to 255 characters) and press ENTER.

The text is added to the library header.

Change Library Comment

This function is used to change the program library comment. This function performs the equivalent action of the **modlib -d** command. This is a Groupware feature and is not available in all versions of the compiler. To change the library comment, follow these steps:

1. Access the Library Menu, select Change Library Comment, and press ENTER. You are prompted:

Comment :

2. Type the new text (up to 255 characters) and press ENTER.

The new text replaces existing extra text in the library header.

Get Last Listing

This function is used to display the results of the last library listing (List Library function). To display the last listing, follow these steps:

1. Access the Library Menu, select Get Last List Listing, and press ENTER.

The results of the last **Library Menu => List Library** execution are displayed in the Review Window.

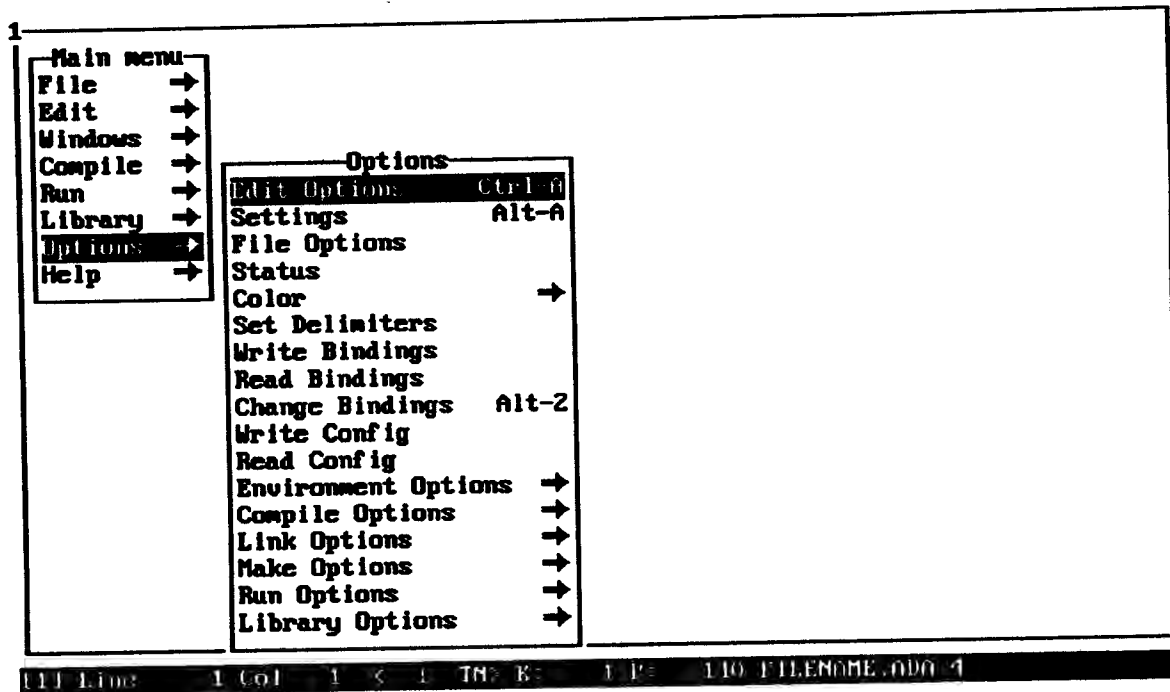
Get Library Info

This function is used to display the results of the last Library Menu selection executed, other than List Library. To display library information, follow these steps:

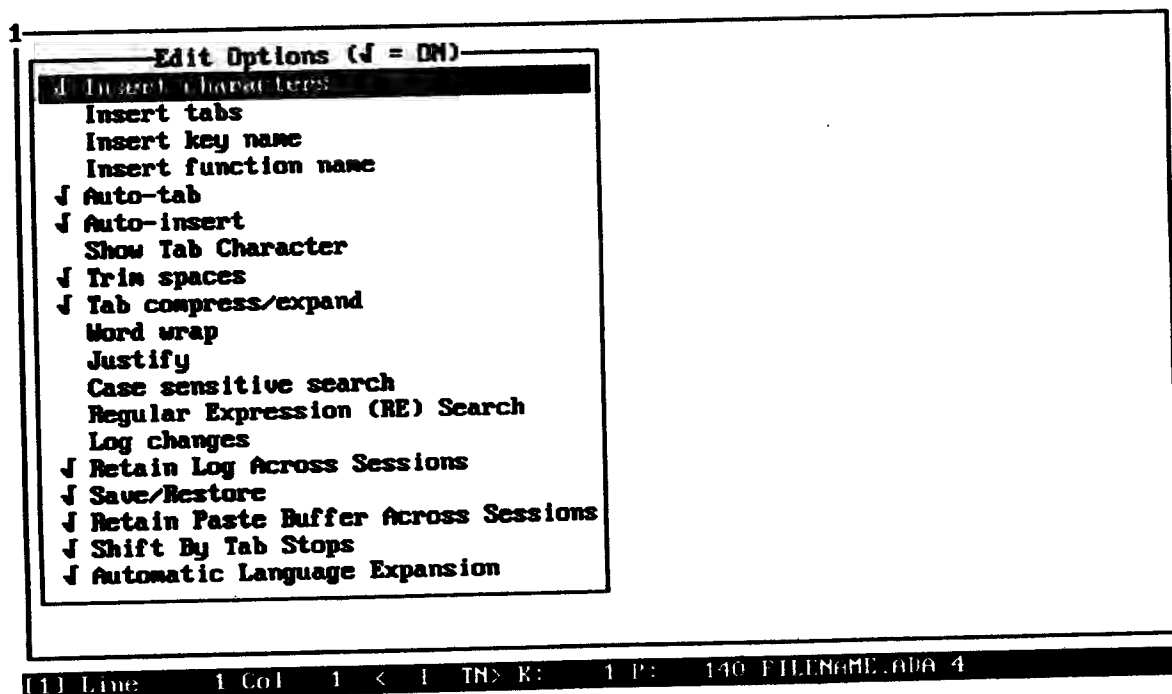
1. Access the Library Menu, select Get Library Info, and press ENTER.

The results of the previous Library Menu selection performed (except for List Library) is displayed in the Review Window. To close the Review Window, press Shift-F2.

4.11 Options Menu



4.11.1 Edit Options (Ctrl-A)



The Edit Options Menu is used to set several editing and text processing options.

To set an option, follow these steps:

1. Access the Edit Options Menu and press **ENTER**. The Edit Options screen appears. Options that are set to On have a check mark located in the margin next to them.
2. Select an option (e.g., Insert Tabs), and press **ENTER**. The setting changes (the check next to the item appears or disappears).
3. Press **ESC** to return to the previous screen.

Menu Selections

Insert Characters

This option determines whether to insert or overstrike characters.

Insert Tabs

This option determines whether the tab function inserts spaces to the next tab location or moves the cursor to the next tab location.

Insert Key Name

This option redefines the keyboard so that the name of the key is inserted (instead of performing the function) when the key is pressed. Type **ESC** to return to the normal behavior.

Insert Function Name

This option redefines the keyboard so that the name of the function associated with a key is inserted when the key is pressed. Type **ESC** to return to the normal behavior.

Auto-Tab

This option determines whether the current line indentation is used or not when a new line is inserted.

Auto-Insert

This option determines whether pressing the **ENTER** (or Return on some systems) key inserts a blank line or not when it moves the cursor to the next line.

Show Tab Character

This option determines whether the tab character is displayed as is or expanded into white space.

Trim Spaces

This option determines whether or not trailing spaces are trimmed.

Tab Compress/Expand

This option determines whether tabs are compressed or expanded during file processing.

Word Wrap

This option determines whether words are wrapped or not when they go beyond the right margin.

Justify

This option determines whether right justification is performed or not when words are wrapped and paragraphs are reformatted.

Case Sensitive Search

This option determines whether exact case match (e.g., upper case) is considered or not when searching occurs.

Regular Expression (RE) Search

This option determines whether pattern matching or normal searching occurs.

ACE Features

Log Changes

This option determines whether a log for file changes is maintained or not.

When this option is on, ACE will maintain a log of all changes made to a file. Insert File and Reread File cause ACE to start logging over. This allows you to undo your file modifications change by change to a previous state. This is not an undo facility or journaling facility, but a reverse file changes facility.

Most undo facilities keep track of your last n operations (file changes and others) for the current editing session. In fact, changing to a different window usually causes undo logging to start over again.

A journaling system typically records keystrokes. For playback in the event of system crash. The ACE Journaling facility provides this feature.

The log retains all changes made to a file, regardless of what window you are working in and across editing sessions. This means that you can change several files, recompile them, relink, test, etc., and then reverse the changes made to each file. It serves as a simple integrated version management system.

When logging is on, all file changes are saved in the subdirectory **ACELOG** which is automatically created in the current file's directory. If the file is in the directory **\ACE**, then log changes will be stored in **\ACE\ACELOG**.

Logging does not seem to significantly slow down ACE, in fact, it is hardly noticeable.

The log files can grow quite large when significant changes are made to a file. You will want to clean out these logging subdirectories, when they begin to get large. When you have a stable version of the file, delete the corresponding log file. For example, if your file is called **\ACE\X.X**, then delete the log file using the command: **del \ACE\ACELOG\X.X**. To delete all the log files use: **del \<dir>\ACE-LOG*.***, where **<dir>** is the directory whose log files you want removed. If you remove the **ACELOG** subdirectory, ACE will recreate it to perform logging.

The logging function provides limited consistency checking. For example, if you make some changes with logging, turn logging off and make changes, turn logging back on and make changes, then if you attempt to reverse file changes past the unlogged area, then the message "Log file corrupted" will appear and you will be unable to undo any further changes.

Logging suggestions:

1. Keep logging on or off, to help ensure consistency.
2. Watch your disk space.
3. Logging performs no internal buffering (it relies on DOS), so reversing file changes can be somewhat slow.

To set ACE to automatically generate logs, follow these steps:

1. Access the Log Changes option and press ENTER.

The setting changes. Each time you start ACE a log will be created (until you turn logging off).

If a log is to be retained from one session to another, you also should turn on Retain Log Across Sessions.

Retain Log Across Sessions

This option determines whether the log file is retained across sessions or deleted at the end of the session.

Save/Restore

This option determines whether or not the window/file editing information is saved for restart.

Retain Paste Buffer Across Sessions

This option determines whether the paste buffer is retained across sessions or deleted at the end of the session.

Shift By Tab Stops

This option determines whether or not the shift left and shift right functions move the text to the next tab stop or by the width of the marked text.

Automatic Language Expansion

When this option is off, the Language Symbol Expansion turns off after the first term is expanded and returns you to the current window. See section 3.6 for more information.

4.11.2 Settings

Settings	
Right margin	20
Tab size	8
Line size	1
Change Case Mode	Flip
Insert #	0
Cursor size	start: 6 end: 7
Checkpoint Frequency (0=off)	10 seconds
Mouse scaling	8 mickeys
Screen Rows	25
Logging Granularity	0

111 Line 1 Col 1 < 1 TR2 K: 1 P: 110 FILENAME.ODD 1

The Settings Menu is used to set and display the current values for the right margin, tab size, line size, case mode, insert number, and cursor size.

To change a setting, follow these steps:

1. Access the Settings Menu, select an option (e.g., Line Size), and press ENTER.

You are prompted for the new value, for example:

Line Size:

2. Type the new setting and press ENTER. You are prompted to save the new setting:

Write Changed Settings To Config File?
Yes
No

3. If you accept the new settings, highlight Yes and press ENTER. However, if you do not, highlight No and press ENTER.

You are prompted again:

Write config file [default=<path>\<config file>]:

4. If you accept the file name and path, just press ENTER. If not, type a different one and press ENTER.

The new setting is written to the configuration file, a message appears:

Writing config file

And, you are returned to the previous screen.

Menu Selections

Right Margin

This option sets the right margin for word wrapping and paragraph reformatting.

Tab Size

This option sets the size of tabs for compression and expansion (the default is 8). When ACE finds a tab in the file, this tab size is used to adjust the line. When tab compress/expand is on all tabs are replaced by spaces. This parameter should very rarely, if ever, be changed.

Line Size

This option sets the line size for the line drawing mode. Press Alt-H to see the Line Drawing Help screen which also contains the line size examples and information.

Change Case Mode

The Change Case Mode option determines the case change operation to perform when executing the Change Case and Change Case Marked Lines functions (see section 4.6.2.). The choices are:

- Flip – change case to the opposite of its current state
- Upper – change to all upper case
- Lower – change to all lower case

Insert #

This option sets the number to be inserted by the Insert Number function. The number is always incremented by one each time it is used.

Cursor Size

This option determines the cursor size.

Defines the start and end cursor line of the cursor. You may need to readjust the cursor size when using some of the extended screen modes (e.g., 43 line mode).

Normal settings are:

- Monochrome: start = 11; end = 13
- Color/CGA: start = 6; end = 7
- EGA: start = 6; end = 7 (through cursor emulation)
- VGA: start = 14; end = 15

Checkpoint Frequency

This option determines the length of time without a key press before ACE saves your latest file changes to disk. To set Checkpoint Frequency, access the Checkpoint Frequency option and press ENTER. ACE prompts you for the number of seconds.

Mouse Scaling

This option determines the rate of mouse-driven cursor speed. The higher the value, the slower the cursor moves.

Screen Rows

This option sets the number of rows for the screen. If supported by your system you can change the number of rows to 25, 43, or 50. To set the number of Screen Rows, access the Screen Rows option and press ENTER. You are prompted for the number of lines. The current value is shown in brackets ([]).

Logging Granularity

This option sets the logging granularity. It is only meaningful if logging is on.

4.11.3 File Options

```

1
File Options (J = ON)
Ignore Extraneous EOFs
Break Long Lines
J Delete 0 Length Files
J Backup
J Retain Backup Across Sessions
111 Line 1 Col 1 < 1 TN> R: 1 P: 140 FILENAME ADD 4

```

This menu is used to set several file options. A check mark located next to the item indicates the option is turned on.

To set a File Option, follow these steps:

1. Access the File Options Menu, select an option (e.g., Break Long Lines), and press ENTER. The setting changes (the check next to the item appears or disappears).
2. Press ESC to return from the File Options Menu.

Menu Selections**Ignore Extraneous EOFs**

This option determines whether or not DOS end-of-file characters (Ctrl-Z's), which appear before the physical end of the file are ignored in file processing operations. When the switch is on, file reading does not stop at an extraneous EOF.

ACE Features

Break Long Lines

This option determines whether or not ACE breaks long lines, moving the excess length down to the next line. If this switch is off, the excess is deleted.

Delete 0 Length Files

This option determines whether empty files (0 length) created during an editing session are deleted or not at the end of the session.

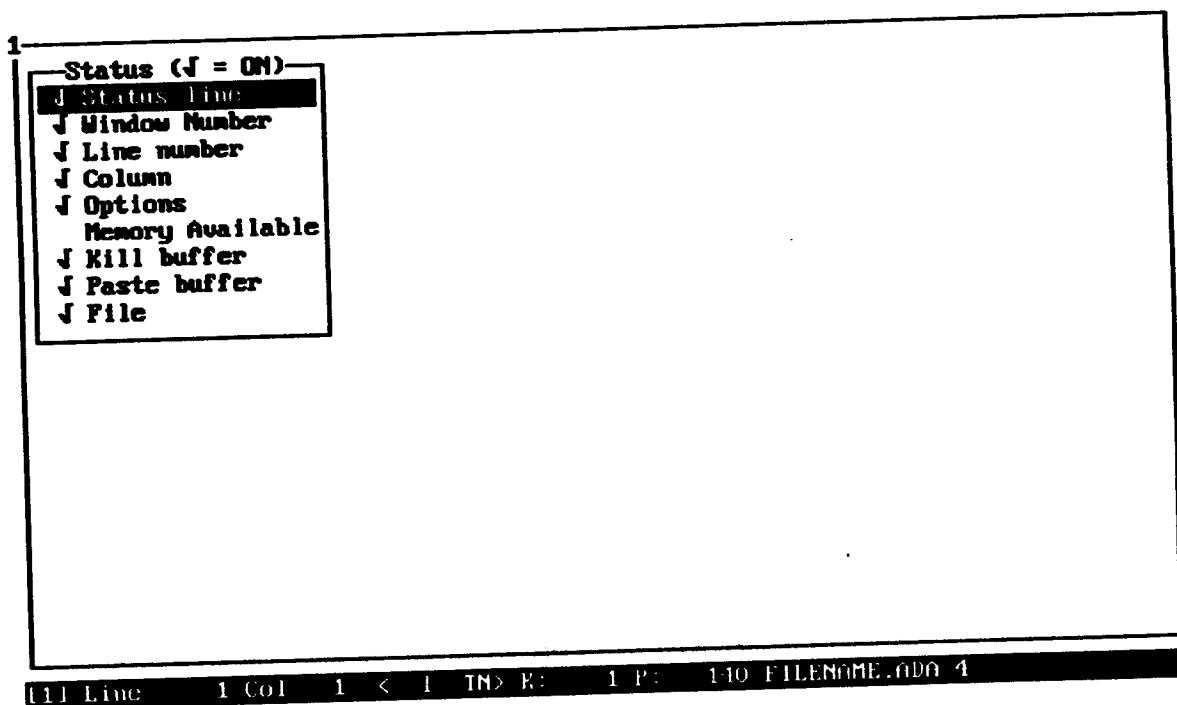
Backup

This option determines whether or not an automatic backup of the file is created.

Retain Backup Across Sessions

This option determines whether the backup file is deleted or not at the end of the editing session.

4.11.4 Status



The Status Menu is used to set the type of information displayed on the status line.

To set the Status display for any option, follow these steps:

1. Access the Status Menu, select an option (e.g., Status Line), and press ENTER.

The highlighted option is set and the Yes/No value changes.

For a Status Line example and more information, see section 2.3.

Menu Selections

Status Line

This option turns on/off the status line display at the bottom of the screen.

Window Number

This option determines whether or not window number data is included in the status line display.

Line Number

This option determines whether or not line number data is included in the status line display.

Column

This option determines whether or not column data is included in the status line display.

Options

This option determines whether or not option data is included in the status line display.

Memory Available

This option determines whether or not available memory data is included in the status line display.

Kill Buffer

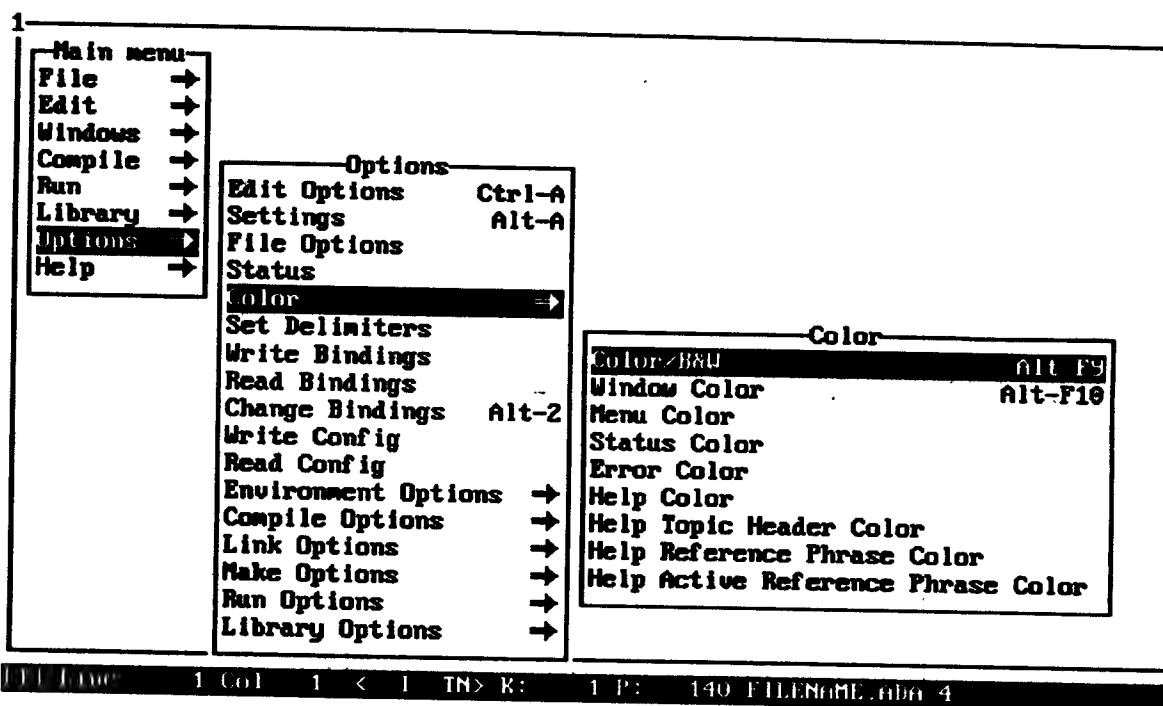
This option determines whether or not kill buffer data is included in the status line display.

Paste Buffer

This option determines whether or not paste buffer data is included in the status line display.

File

This option determines whether or not file data is included in the status line.

4.11.5 Color Menu

This menu is used to set the color for various screen displays (i.e., windows, menu displays, error and status messages, and help screens). To set the color you must select a foreground and background color when making changes to a screen color. To change a screen color, follow the directions below.

1. Access the Select Color Screen and use the up/down arrow keys to highlight "Foreground" (or "Background").
2. Press the left or right arrow key until the foreground (or background) color you want appears. The colors next to each display change to show the selection as you toggle through the colors. Press ENTER. You are prompted:

```
Write Changed Settings To Config File?
Yes
No
```

3. Highlight Yes or No and press ENTER. You are prompted again:

```
Write config file [default=<path>\<config file>]:
```

4. If you accept the configuration file (and path) designation, press ENTER. If you want to specify a different file and/or path, type it and press ENTER.

The foreground (or background) color setting is changed in the configuration file and you are returned to the current window. The screen displays the new foreground (or background) color.

If your screen is blinking, it means you have selected one of the blinking colors. To stop your screen from blinking, return to the Select Color screen and select a non-blinking color.

Menu Selections

Color/B&W (Alt-F9)

This function turns off color, allowing you to use ACE with a monochrome monitor.

This function is only used if you are running ACE on a PC with a color/graphics adapter and a monochrome monitor. In this case, you will have to reconfigure ACE, turning off the color. Follow these steps:

1. Access the Color Menu, select Color/B&W, and press ENTER.

The Color Menu disappears and the current window appears in black and white.

Window Color (Alt-F10)

This function accesses the Select Color Screen.

Menu Color

This function accesses the Select Color screen used to set color for menus.

Status Color

This function accesses the Select Color screen used to set foreground and background color for the status line.

Error Color

This function accesses the Select Color screen used to set color for error messages.

Help Color

This function accesses the Select Color screen used to set background color for Help displays.

Help Topic Header Color

This function accesses the Select Color screen used to set the general color for the highlighted topic header in online Help displays.

Help Reference Phrase Color

This function accesses the Select Color screen used to set the general color for the highlighted reference phrase in online Help displays.

Help Active Reference Phrase Color

This function accesses the Select Color screen used to set color for the currently active help reference phrase.

4.11.6 Set Delimiters

This function sets word delimiting characters (Right Word, Left Word, Delete Word, and delimited function types). Pressing ENTER toggles the delimiter on/off.

4.11.7 Write Bindings

This function writes the key bindings to a file.

4.11.8 Read Bindings

This function reads a key binding file.

4.11.9 Change Bindings (Alt-2)

Function -> Key Binding	
(None)	
ASCII Chart	Ctrl-C
Abort	Alt-Z
Add Library Link	
Augment Library	
Backspace	Backspace
Backspace And Wrap	Ctrl-Backspace
Backup File	
Balance {}[]()	Alt-1
Bind Macro To Key	
Break And Next Line	Alt-J
Break Line	Ctrl-K
Bullet	Ctrl-B
Bullet Off	Alt-B
Center Line	Alt-C
Change Bindings	Alt-Z
Change Case	Alt-T
Change Case Marked Lines	
Change Internal Library Name	
Change Library Comment	
Change Library Link	
Close Window	Shift-F2

Esc-cancel; Enter-Bind function; Ctrl-Enter>Create Key Map; Ctrl-D-Del Key Map

The entries in this menu represent ACE functions whose key bindings can be changed. Each of these menu entries is used to reset the key binding of the function with the same name. For example, the "Frame/Move" entry is used to reset the key binding of that function from Alt-F8 to another setting.

WARNING

Although "Change Bindings" is provided with ACE, this function should be used carefully. A change in one key binding may unexpectedly affect another function. That is, if you enter a key binding already assigned, that key binding assigned is deleted and reassigned to the selected function. No error or warning message is given. You can also assign to multiple keys the same function. To disable a key, assign it to the Null function.

To change a Function → Key Binding, follow these steps:

1. Access the Function → Key Binding Menu, select the function you wish to assign a key binding, and press ENTER.

You are prompted for a new binding. For example, for Add Library Link, the following prompt appears:

```
Bind Add Library Link to:
```

2. Type in the new binding (the exact keys to which you want the function assigned) and press ENTER. The Function → Key Binding screen is redisplayed.
3. When you press ESC to exit this screen. You are prompted:

```
Write Bindings to file?  
Yes  
No
```

4. Select Yes or No.
5. If you select Yes, you are further prompted:

```
Write key binding file [default=ace.key]:  
Yes  
No
```

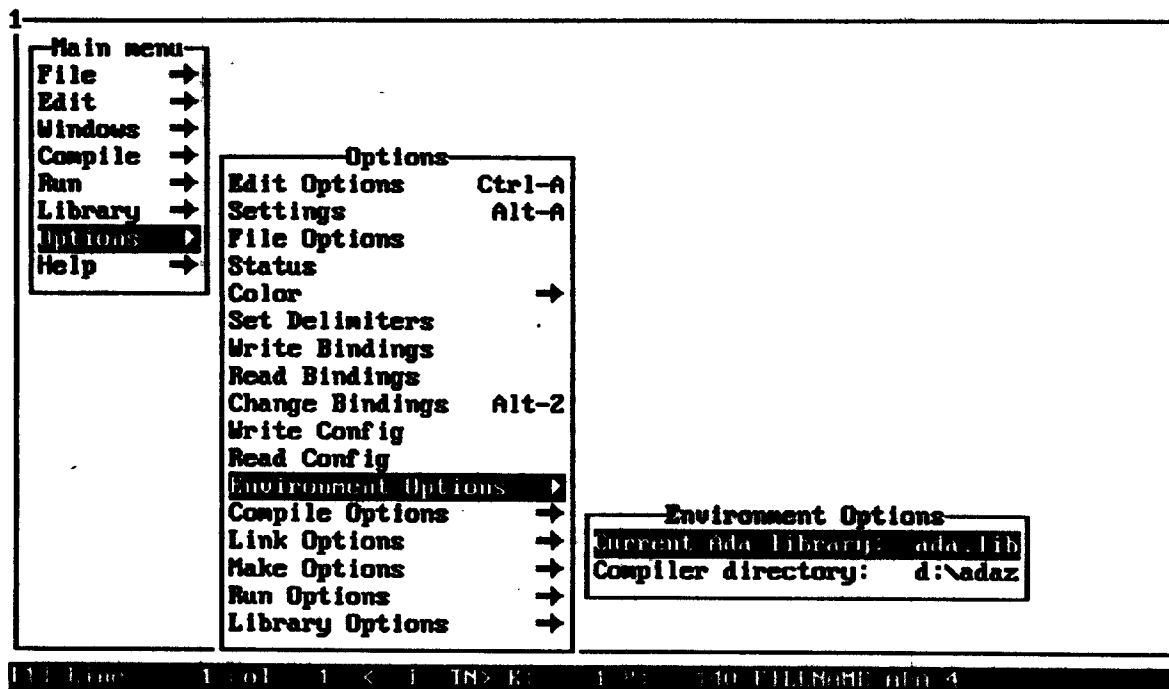
4.11.10 Write Config

This function writes the options and settings to a configuration file. The default file is shown.

4.11.11 Read Config

This function is used to read the current options and settings from the configuration file.

4.11.12 Environment Options Menu



The Environment Options Menu is used to set the Ada program development environment.

To change Environment options, follow these steps:

1. Access the Environment Options Menu, select an option (e.g., Current Ada Library), and press **ENTER**.

You are prompted for input (e.g. Library Name:).

2. Type the requested information and press **ENTER**.

The option is set and the new name appears in the highlighted option area.

Menu Selections

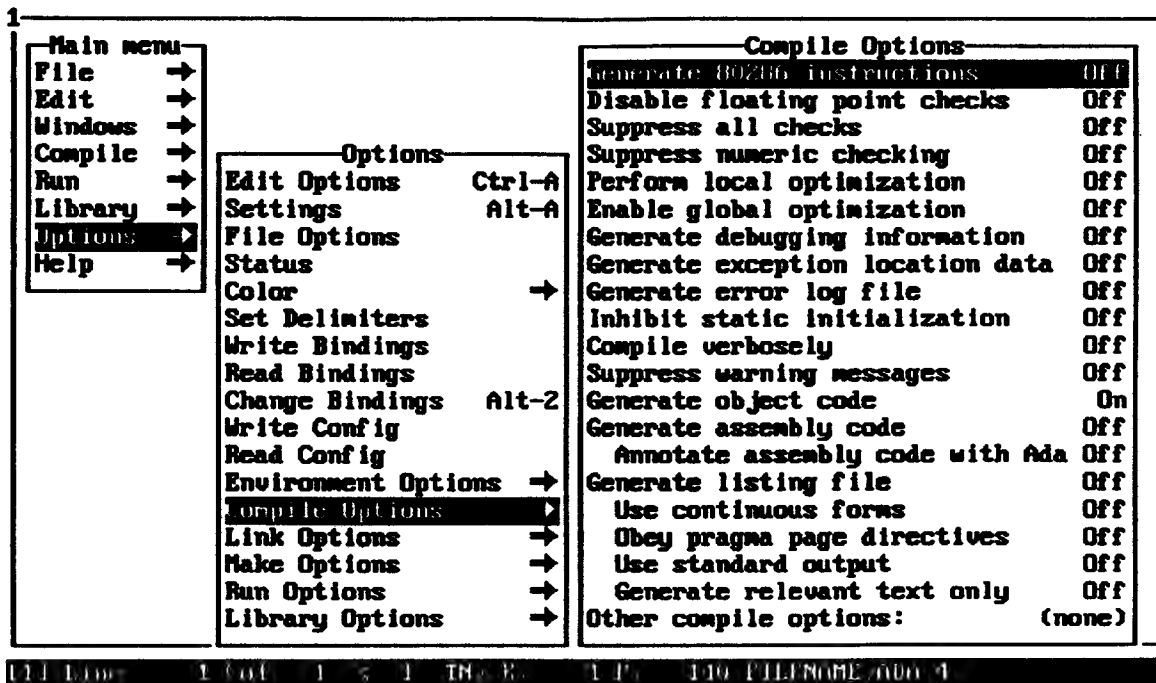
Current Ada Library

This option sets the current Ada program library. This option performs the equivalent action of the **ada -L** command-line option. For more information, see the *Meridian Ada Compiler User's Guide*.

Compiler Directory

This option displays the current compiler directory. It cannot be changed directly. To change it you must change the DOS PATH environment variable.

4.11.13 Compile Options Menu



The Compile Options Menu is used to set the compilation options (ada command options). The ada command along with its options are described in detail in the *Meridian Ada Compiler User's Guide*. Please refer to that manual for information on how and when to use these options.

To set compilation options, follow these steps:

1. Access the Compile Options Menu, select the option (e.g., Generate 80286 Instructions), and press ENTER. Pressing ENTER toggles the on/off switch.
2. When the option is set to the desired value, simply back out of the menu using the ESC key.

Menu Selections

Generate 80286 Instructions

When this option is set to On, instructions are generated for a 80286 target. This option performs the equivalent action of the **ada -fS** command-line option.

Disable Floating Point Checks

When this option is set to On, floating point checks for a math co-processor are suppressed before the math co-processor instructions, resulting in a smaller and faster program. This option performs the equivalent action of the **ada -fF** command-line option.

Suppress All Checks

When this option is set to On, all automatic checking (including numeric checking) is suppressed. This reduces the size of the code, creating a "production quality" program with better performance. This option performs the equivalent action of the **ada -fs** command-line option.

Suppress Numeric Checking

When this option is set to On, numeric checking (**division_check** and **overflow_check**) is suppressed. This option performs the equivalent action of the **ada -fN** command-line option.

Perform Local Optimization

When this option is set to On, the compiler runs an additional optimization pass. This optimization removes common subexpressions, dead code, unnecessary jumps, and performs loop optimization. This option performs the equivalent action of the **ada -g** command-line option.

Enable Global Optimization

When this option is set to On, the internal form file is saved, permitting subsequent global optimization at link time. This option performs the equivalent action of the **ada -K** command-line option.

Generate Debugging Information

When this option is set to On, the compiler generates appropriate code and data for debugging operations. This option performs the equivalent action of the **ada -fD** command-line option.

Generate Exception Location Data

When this option is set to On, exception location information (source file names and line numbers) is maintained for internal checks. This option performs the equivalent action of the **ada -fL** command-line option.

Generate Error Log File

When this option is set to On, the compiler generates a log file containing all error and warning messages produced during the compilation. This option performs the equivalent action of the **ada -fE** command-line option.

Inhibit Static Initialization

When this option is set to On, static initialization is suppressed for variables (but not for constants) and assignments to each variable component are performed in the code. This option performs the equivalent action of the **ada -fR** command-line option.

Compile Verbosely

When this option is set to On, the compiler prints the name of each subprogram, package, or generic as it is compiled. This option performs the equivalent action of the **ada -fv** command-line option.

Suppress Warning Messages

When this option is set to On, the compiler does not print warning messages about ignored pragmas, significant exceptions, or other potential problems. This option performs the equivalent action of the **ada -fw** command-line option.

Generate Object Code

When this option is set to On, object code is generated.

Generate Assembly Code

When this option is set to On, the code generator produces an assembly language source file and halts further processing. This option performs the equivalent action of the **ada -S** command-line option.

Annotate Assembly Code with Ada

When this option is set to On, the compiler annotates an assembly language output file. The output is accompanied by comments containing the Ada source statements corresponding to the assembly language code sections written by the code generator. This option performs the equivalent action of the **ada -fa** command-line option.

Generate Listing File

When this option is set to On, the compiler generates a listing file. This option performs the equivalent action of the **ada -l** command-line option.

To set ACE to automatically create a compilation listing, follow these steps:

- 1. Access the Compile Options Menu, select Generate Listing File, and press ENTER.**

Now, determine which additional listing options apply and also select them (e.g., Use continuous forms).

The highlighted option is set and the On/Off value changes. ACE will automatically produce a listing, during compilation.

Use Continuous Forms

When this option is set to On, the listing file is a continuous forms output. This option performs the equivalent action of the `ada -lc` command-line option.

Obey Pragma Page Directives

When this option is set to On, the listing file obeys pragma page directives. This option performs the equivalent action of the `ada -lp` command-line option.

Use Standard Output

When this option is set to On, the listing file is sent to the standard output. This option performs the equivalent action of the **ada -ls** command-line option.

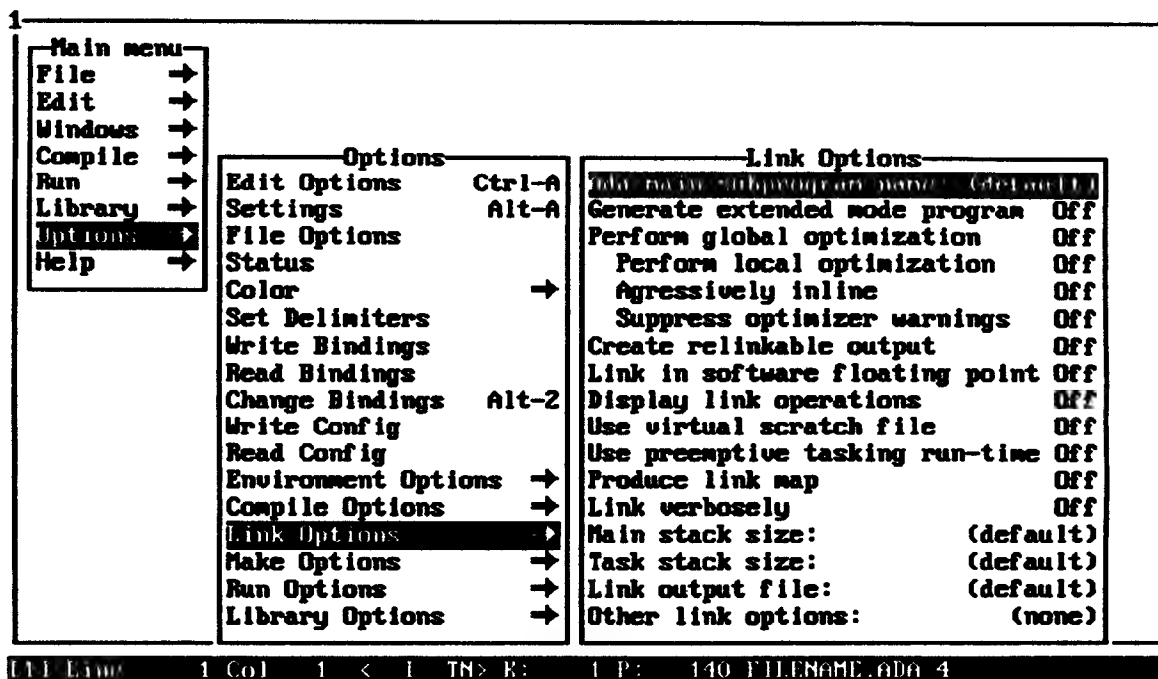
Generate Relevant Text Only

When this option is set to On, the listing file only includes relevant text. This option performs the equivalent action of the `ada -lt` command-line option.

Other Compiler Options

This option is used to specify other, rarely used, compiler options, besides those listed above. See the *Meridian Ada Compiler User's Guide* for the other options.

4.11.14 Link Options Menu



This menu is used to set link options (**bamp** command options). The **bamp** command, along with its options, are described in detail in the *Meridian Ada Compiler User's Guide*. Please refer to that manual for information on how and when to use these options.

To set fill-in options, follow these steps:

1. Access the Link Options Menu, select the fill in option (e.g., Ada Main Program Unit Name), and press ENTER.

You are prompted for information (e.g., Main Program Unit:).

2. Type in the requested information and press ENTER.

The option is set and the new information appears in the highlighted option area.

To set On/Off options, follow these steps:

1. Access the Link Options Menu, select the On/Off option (e.g., Generate Extended Mode Program), and press ENTER. Pressing ENTER toggles the On/Off switch.
2. When the switch is set to the desired value, simply back out of the menu using the ESC key.

Menu Selections

Ada Main Subprogram Name

When this option is set for the default, the most recently compiled compilation unit which satisfies the Ada main program profile (that is, a parameterless procedure) is used. When a program unit name is supplied, the default is overridden.

Generate Extended Mode Program

When this option is set at On, an extended mode program (.exp file) is produced that may be run with the **ramp** command in Extended Mode. If this option is set to No, a real mode program (.exe file) is produced. This option performs the equivalent action of the **bamp -x** command-line option. For more information, see the *Meridian Ada Compiler User's Guide*. This is an extended mode feature and is not available in all versions of the compiler.

Perform Global Optimization

When this option is set at On, the linker performs global optimization on the program. This option performs the equivalent action of the **bamp -g** command-line option.

Perform Local Optimization

When this option is set at On, the linker performs global and local optimization on the program. This option performs the equivalent action of the **bamp -G** command-line option.

Aggressively Inline

When this option is set at On, aggressive inlining is performed on the program when it is optimized. This option performs the equivalent action of the **bamp -A** command-line option.

Suppress Optimizer Warnings

When this option is set at On, optimizer warnings are suppressed during the linking process. This option performs the equivalent action of the **bamp -W** command-line option.

Create Relinkable Output

When this option is set at On, an object file (.obj file) is produced, instead of an executable file (.exe file). This option performs the equivalent action of the **bamp -x** command-line option.

Link in Software Floating Point

When this option is set at On, a program containing floating point computations can be run with or without a math co-processor chip. This option performs the equivalent action of the **bamp -u** command-line option.

Display Link Operations

When this option is set at On, the linker produces a printout identifying operations performed during the creation of the executable program. This option performs the equivalent action of the **bamp -P** command-line option.

Use Virtual Scratch File

When this option is set at On, larger programs can be linked. The object files are linked in a scratch file created by this option. This option performs the equivalent action of the **bamp -V** command-line option.

Use Preemptive Tasking Run-Time

When this option is set at On, the program is linked with a version of the tasking run-time that supports preemptive task scheduling. This option performs the equivalent action of the **bamp -I** command-line option.

Produce Link Map

When this option is set at On, the linker produces a text file containing a link map is written. This option performs the equivalent action of the **bamp -m** command-line option.

Link Verbosely

When this option is set at On, the linker produces a printout identifying the operations it performs in building the main program (e.g., name of the program library checked). This option performs the equivalent action of the **bamp -v** command-line option.

Main Stack Size:

This option is used to set the main program stack size (number of decimal bytes). This option performs the equivalent action of the **bamp -M** command-line option.

Task Stack Size:

This option is used to set the stack size (number of decimal bytes) allocated to all tasks to be activated by the Ada program. This option performs the equivalent action of the **bamp -s** command-line option.

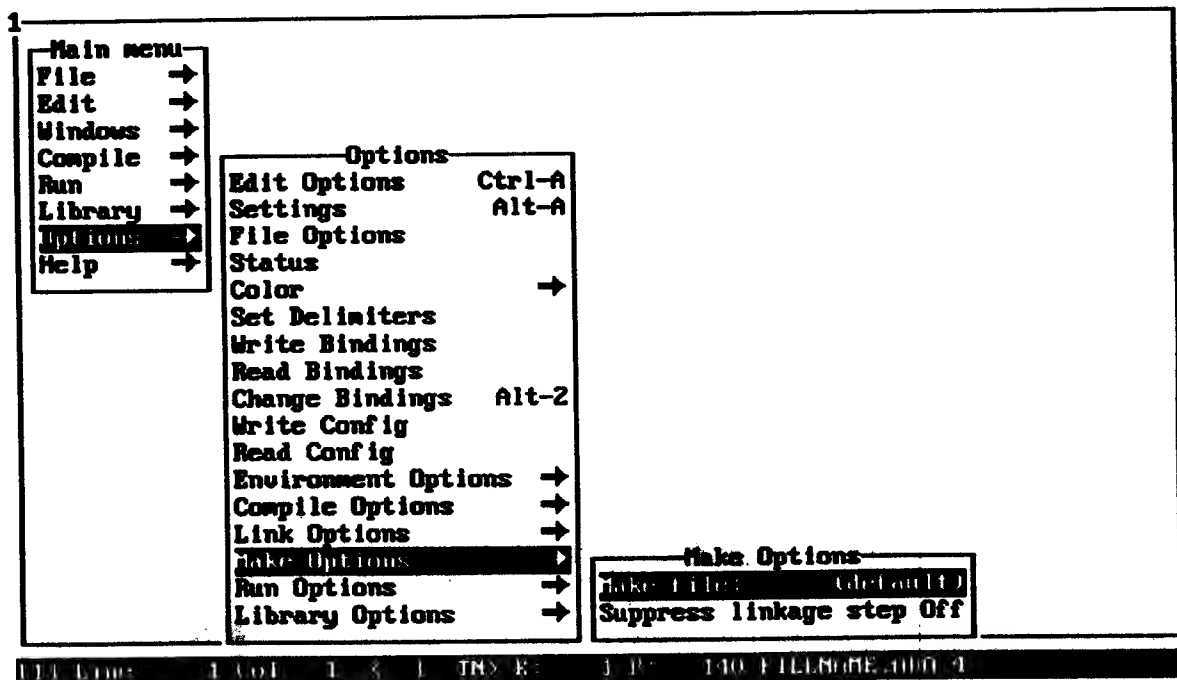
Link Output File:

This option is used to specify an alternate executable-file output name. This option overrides the default output file name. This option performs the equivalent action of the **bamp -o** command-line option.

Other Link Options:

This option can be used to specify other, rarely used, linker options besides those provide above. For more information, see the *Meridian Ada Compiler User's Guide*.

4.11.15 Make Options Menu



This menu is used to set the options used by the Make Program function (**amake** command options). The **amake** command, along with its options, are described in detail in the *Meridian Ada Compiler User's Guide*. Please refer to that manual for information on how and when to use these options.

Menu Selections

Make File

This option is used to identify an alternate make file name. This option performs the equivalent action of the **amake -M** command-line option. To set Make Program options, follow the steps provided below.

1. Access the Make Options Menu, select Make File, and press ENTER.

You are prompted, "Make File Name".

2. Type the Make File name and press ENTER.

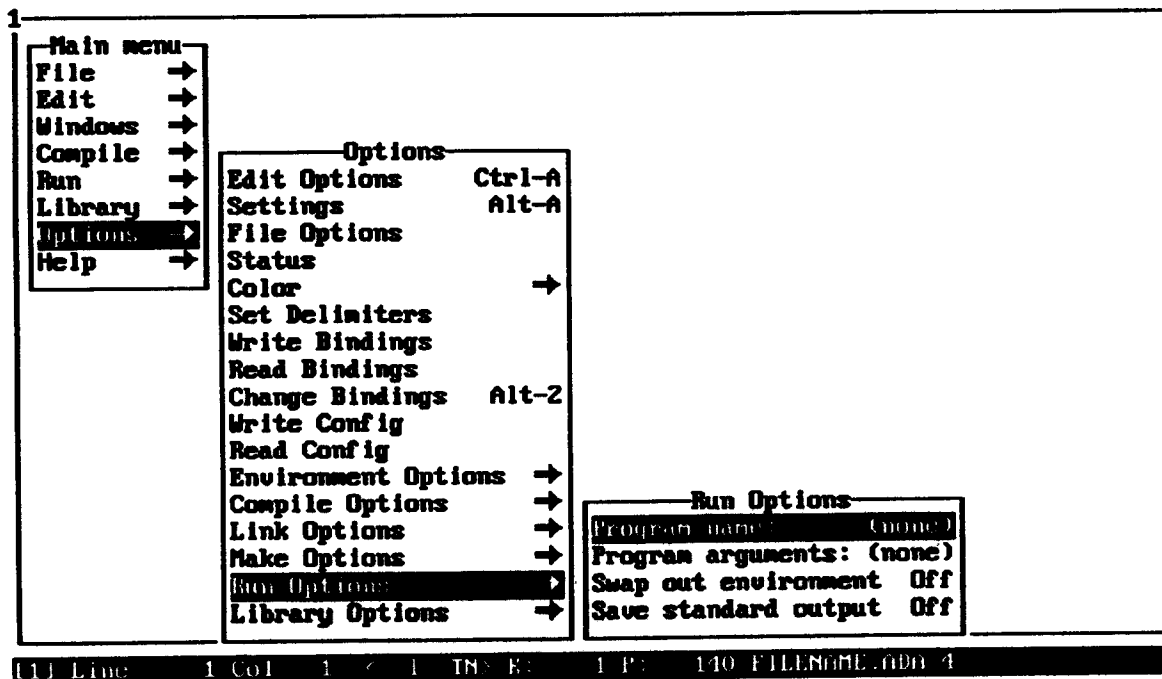
The option is set and the new name appears in the highlighted option area.

Suppress Linkage Step

When this option is set at On, there is no linkage step after compilation. This option performs the equivalent action of the **amake -n** command-line option.

1. Access the Make Options Menu, select Suppress Linkage Step, and press ENTER. Pressing ENTER toggles the On/Off switch.
2. When the switch is set to the desired value, simply back out of the menu using the ESC key.

4.11.16 Run Options



The Run Options Menu is used to set options used by the Run Program function. These options should be set before Run Program is used to execute the program.

Menu Selections

Program Name

This option is used to identify the name of the program to run.

1. Access the Run Options Menu, select Program Name, and press ENTER.

You are prompted for information.

2. Type the requested information and press ENTER.

The option is set and the new information appears in the highlighted option area.

Program Arguments

This option is used to set the command line arguments for the program run.

1. Access the Run Options Menu, select Program Arguments, and press ENTER.

You are prompted for information.

2. Type the requested information and press ENTER.

The option is set and the new information appears in the highlighted option area.

Swap Out Environment

When this option is set at On, the ACE environment is swapped out while the program is executing.

1. Access the Run Options Menu, select Swap Out Environment, and press ENTER. Pressing ENTER toggles the On/Off switch.

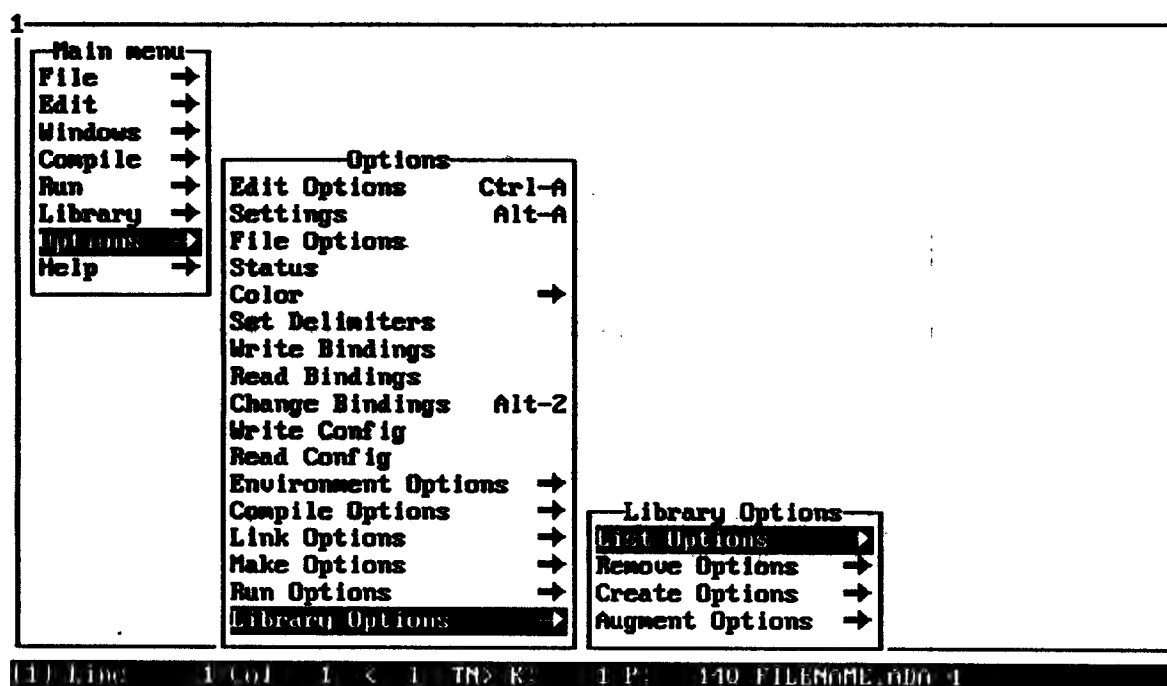
2. When the option is set to the desired value, simply back out of the menu using the ESC key.

Save Standard Output

When this option is set at On, the standard output of the program run is saved in a file. This information may be reviewed with the Get Run Info command.

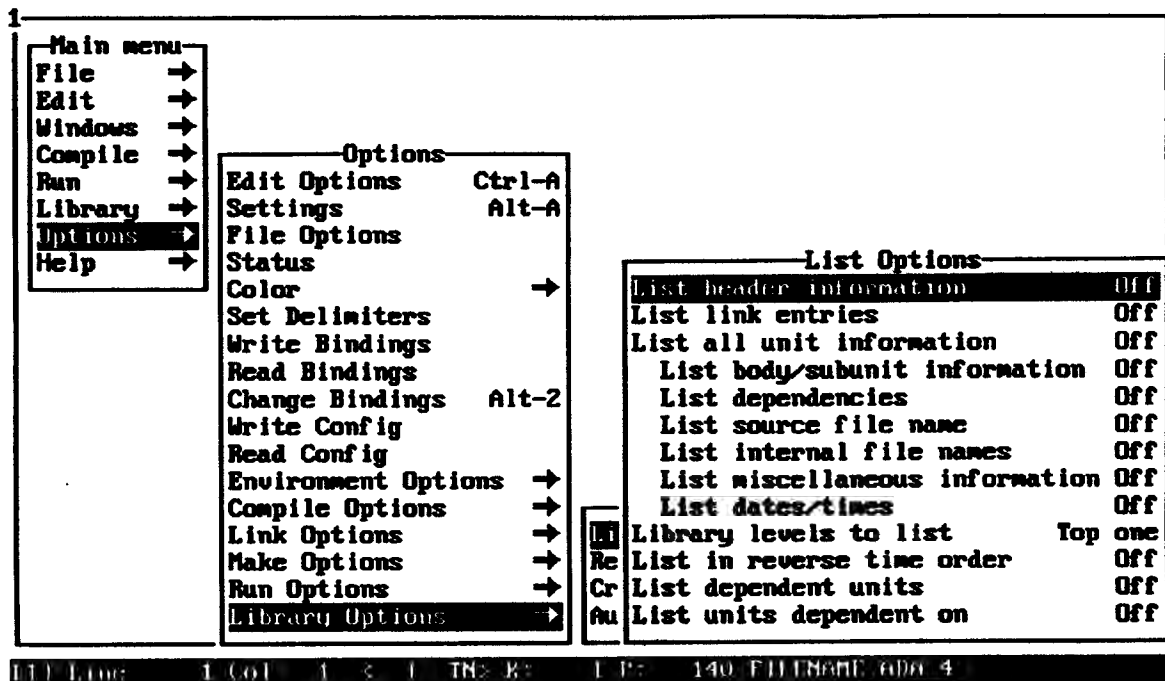
1. Access the Run Options Menu, select Save Standard Output, and press ENTER. Pressing ENTER toggles the On/Off switch.
2. When the switch is set to the desired value, simply back out of the menu using the ESC key.

4.11.17 Library Options



The Library Options Menu is used to access Create Options, Remove Options, Augment Options, and List Options submenus.

4.11.17.1 List Options



The List Options Menu lists those options that affect the behavior of the List Library function (**lslib** command). The **lslib** command, along with its options, are described in detail in the *Meridian Ada Compiler User's Guide*. Please refer to that manual for information on how and when to use these options.

To set any list option, access the List Options Menu, select an option (e.g., List Header Information), and press ENTER. The highlighted option is set and the On/Off value changes.

Menu Selections

List Header Information

When this option is set at On, the listing contains header information (i.e., library version stamp, target machine type, library creation time, etc.). This option performs the equivalent action of the **lslib -h** command-line option.

List Link Entries

When this option is set at On, the listing identifies all libraries link entries in the library. This option performs the equivalent action of the **lslib -k** command-line option.

List All Unit Information

When this option is set at On, the listing is in the long (extensive) format, including the source file name, library entry time, library update time, and list of dependencies. This option performs the equivalent action of the **lslib -l** command-line option.

List Body/Subunit Information

When this option is set at On, the listing identifies the body and subunits if any. This option performs the equivalent action of the **lslib -o b** command-line option.

List Dependencies

When this option is set at On, the listing includes the dependencies. This option performs the equivalent action of the **lslib -o d** command-line option.

List Source File Name

When this option is set at On, the listing includes the source file name. This option performs the equivalent action of the **lslib -o f** command-line option.

List Internal File Names

When this option is set at On, the listing includes the compiler generated internal link symbol name and the base file name used to generate internal files such as the object file. This option performs the equivalent action of the **lslib -o i** command-line option.

List Miscellaneous Information

When this option is set at On, the listing includes the unit type (i.e., subprogram, main subprogram, package, or task body) and indicates if the unit needs recompilation, if it uses tasking or initialization code, and if it uses debugging code. Additionally, the listing includes any link flags added to the unit with the Augment Library function. This option performs the equivalent action of the **lslib -o m** command-line option.

List Dates/Times

When this option is set at On, the listing indicates when the library and its individual units were initially created and last updated. This option performs the equivalent action of the **lslib -o t** command-line option.

Library Levels to List

When this option is set to "Top one", only the units in the current library are listed. When it is set to "Top two", those units as well as the units in the libraries which the current library is directly linked to are listed. This is the level to which the compiler looks to resolve references to units in a context clause. If this option is set to "All", then all units in all libraries accessible to the current library are listed no matter how many library links must be followed. This is the level to which the linker looks to find all of the object files that it needs to build a main program. This option performs the equivalent actions of the **lslib -a** or **lslib -A** command-line options.

List in Reverse Time Order

When this option is set at On, the listing includes units in reverse temporal order (i.e., most recently updated to least recently updated). This option performs the equivalent action of the **lslib -t** command-line option.

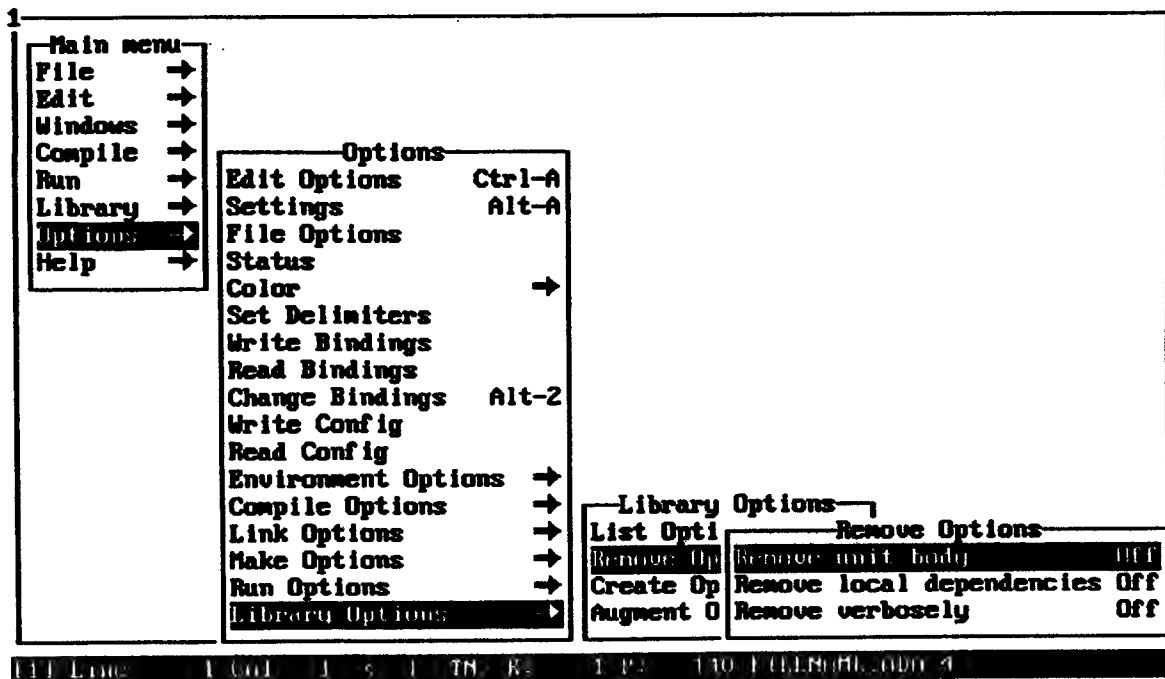
List Dependent Units

When this option is set at On, the listing includes all units on which the specified unit depends. This option performs the equivalent action of the **lslib -d** command-line option.

List Units Dependent On

When this option is set at On, the listing includes everything that depends upon the specified unit. This option performs the equivalent action of the **lslib -w** command-line option.

4.11.17.2 Remove Options



The Remove Options Menu lists those options that affect the behavior of the Remove Library Unit function (**rmLib** command). The **rmLib** command, along with its options, are described in detail in the *Meridian Ada Compiler User's Guide*. Please refer to that manual for information on how and when to use these options.

To change a remove option, follow these steps:

1. Access the Remove Options Menu, select an option (e.g., Remove Local Dependencies), and press ENTER. Pressing ENTER toggles the On/Off switch.
2. When the switch is set to the desired value, simply back out of the menu using the ESC key.

Menu Selections

Remove Unit Body

When this option is set at On, only information about the body of the unit is deleted. This option performs the equivalent action of the **rmLib -b** command-line option.

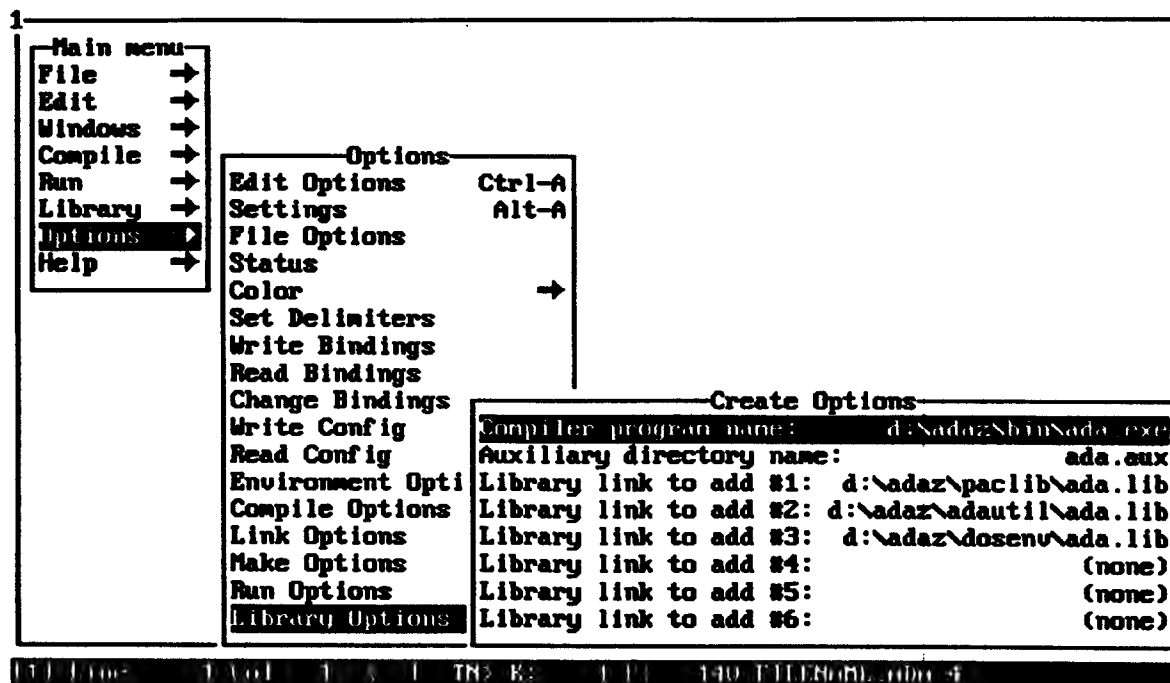
Remove Local Dependencies

When this option is set at On, all local entries on which the unit depends (including the unit entry itself) are deleted. This option performs the equivalent action of the **rmLib -x** command-line option.

Remove Verbosely

When this option is set at On, a listing is produced, identifying each file as it is removed. This option performs the equivalent action of the **rmLib -v** command-line option.

4.11.17.3 Create Options



The Create Options Menu lists those options that affect the behavior of the Create New Library function (`mklib` and `lnlib` commands). The `mklib` command, along with its options, are described in detail in the *Meridian Ada Compiler User's Guide*. Please refer to that manual for information on how and when to use these options.

To set library creation options, follow these steps:

1. Access the Create Options Menu, select the option (e.g., Library Link to Add #1), and press ENTER.
You are prompted to identify the item (e.g., a program library).
2. Type the appropriate name and press ENTER.
The option is set and the new name appears in the highlighted option area.

Menu Selections

Compiler Program Name

This option identifies the compiler program name. The default is the program `ada.exe` found by searching through the directories in the current PATH environment variable. You should not need to change this option. This option performs the equivalent action of the `mklib -c` command-line option.

Auxiliary Directory Name

This option identifies the auxiliary directory to be associated with the program library. All files created by compilations related to the library will be stored in this auxiliary directory. The default is `ada.aux`. This option performs the equivalent action of the `mklib -a` command-line option.

Library Link to Add #1

This option establishes a link to the specified library. The Standard Library is the default for this option.

Library Link to Add #2

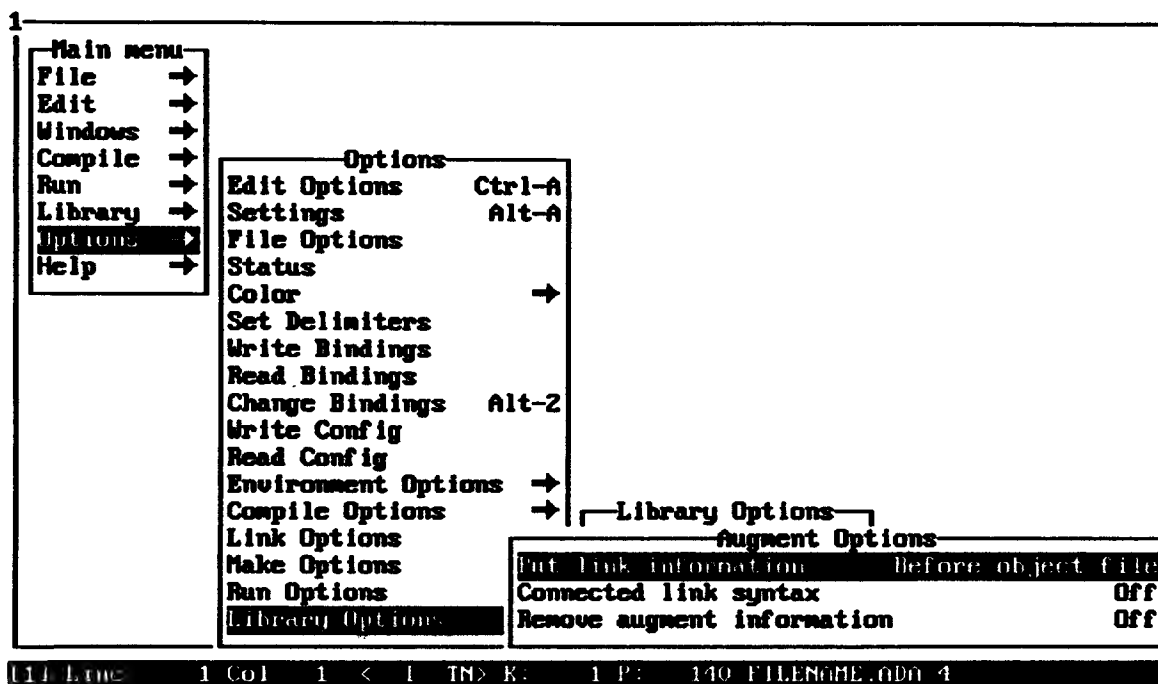
This option establishes a link to the specified library. The Ada Utility Library is the default for this option.

Library Link to Add #3

This option establishes a link to the specified library. The DOS Environment Library is the default for this option.

Library Link to Add #4 – #6

These options establish links to the specified libraries. These are optional, user-defined libraries.

4.11.17.4 Augment Options

The Augment Options Menu lists those options that affect the behavior of the Augment Library function (**auglib** command). The **auglib** command, along with its options, are described in detail in the *Meridian Ada Compiler User's Guide*. Please refer to that manual for information on how and when to use these options.

Examples of types of information that can be attached include some linkage options and the names of non-Ada object files or code libraries.

Menu Options**Put Link Information**

You have three choices. To toggle between the three choices, press ENTER. When the desired option appears, press ESC. Pressing ESC invokes the selected option and returns you to the previous menu.

When this option is set to "After object file", the linker inserts link parameter information after the name of the object file corresponding to the library unit. This option performs the equivalent action of the **auglib -a** command-line option.

When the option is set to "Before object file", the linker inserts the link parameter information immediately before the name of the object file corresponding to the library unit. This option performs the equivalent action of the **auglib -b** command-line option.

If the option is set to "At end of list", the linker inserts the link parameter information after all other object and run-time files being linked into your program. This option performs the equivalent action of the **auglib -e** command-line option.

Connected Link Syntax

When this option is set at On, the linker concatenates the link parameter information to the object file name (with no spaces between).

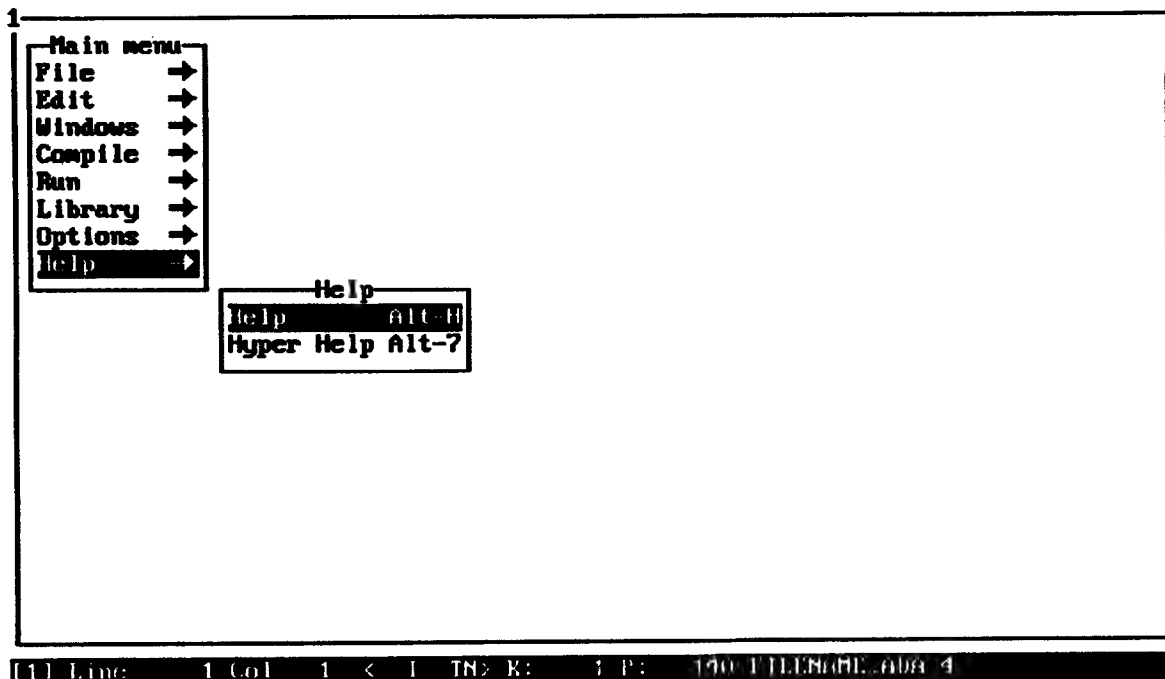
When the option is set at Off, the linker separates the link parameter information from the object file name (no concatenation). This is the default.

This option performs the equivalent action of the **auglib -c** command-line option.

Remove Augment Information

When this option is set at On, link information previously attached to a library unit is removed from the library unit. This option performs the equivalent action of the **auglib -r** command-line option.

4.12 Help Menu



The Help Menu provides access to the on-line help facilities of ACE. The Help function allows you to select ACE specific help. The Hyper Help function allows you access the online Ada Language Reference Manual and DOS Environment and Ada Utility Library documentation.

Menu Selection

Help (Alt-H)

You can get help at virtually any time from inside ACE. The online help covers all ACE functions and features along with compiler specific help.

When you press Alt-H with no menus present, the Help help screen appears. This screen describes how to use the help function. On this screen (and other help screens) you will generally see text that is highlighted. This highlighted text identifies a help topic on which additional information is available. You can select a topic and ACE will display information about it.

Once you are inside a help screen you can scroll through the text using the PgUp, PgDn, and arrow keys. You can also use the arrow keys to select another topic. When you have selected a topic, press ENTER to see the additional information.

You can use the Ctrl-← and Ctrl-→ keys to jump from one topic to another.

To leave any help screen, use the ESC key.

One of the topics on the Help help screen is INDEX. When you select INDEX and press ENTER, ACE displays a complete list of functions and features which have help. You can then select one of these topics and press ENTER to view its help text.

The Help function is also context sensitive. If you select Help while inside a menu, ACE displays help information about the currently highlighted menu item.

ACE often prompts you for information. If you select Help when a prompt is displayed, ACE displays information about that prompt.

Hyper Help (Alt-7)

This function is used to access the online Ada Language Reference Manual and DOS Environment Library and Ada Utility Library documentation. To request Hyper Help, you must be inside a window with no menus present.

When you request Hyper Help, ACE checks your current cursor position for the topic you want to lookup. If your cursor is located on a blank (no topic), ACE displays the Hyper Help contents screen. This screen displays the three online manuals available, and allows you to select which one you want to view.

If you request Hyper Help with your cursor on a single word topic, ACE looks through the Hyper Help database for information about that topic. For example, if your cursor is located on an Ada keyword, the information about that keyword is displayed from the LRM. If your cursor is located on a topic that has more than one definition, you will be prompted with a menu and allowed to select which definition you want to view.

You can also use the Mark Lines function to highlight a multiple word topic and then request Hyper Help. ACE then searches the Hyper Help database for the multiple word topic. This is useful for looking up specific LRM sections such as 3.7.3 or selected subprogram references such as `text_io.put_line`.

See the Help function above for details on selecting and viewing topics within a Hyper Help screen.

Appendix A Key Binding

Function	Key Binding
Abort	Alt-Z
ASCII Chart	Ctrl-C
Backspace	Backspace
Backspace And Wrap	Ctrl-Backspace
Balance () [] { }	Alt-1
Break and Next Line	Alt-J
Break Line	Ctrl-K
Bullet	Ctrl-B
Bullet Off	Alt-B
Center Line	Alt-C
Change Bindings	Alt-2
Change Case	Alt-T
Close Window	Shift-F2
Color/B&W	Alt-F9
Column Left	←
Column Right	→
Copy	F3
Copy & Append	Shift-F3
Cut	F4
Cut & Append	Shift-F4
Define Macro	Ctrl-V
Delete Character	Del
Delete Language Symbol	Alt-L Alt-D
Delete Line	Ctrl-D
Delete Macro	Alt-M
Delete Marked Lines	Alt-D
Delete to End of Line	Ctrl-W
Delete to Next Delimiter	Keypad -
Do Command	Alt-F3
DOS Shell	Alt-F2
DOS Window	Alt-F1

Key Bindings

Function	Key Binding
Edit Options	Ctrl-A
Edit Screen	Alt-5
End of File	Ctrl-PgDn
End of Line	Ctrl-E
End of Screen	END key
Execute Macro	Ctrl-X
Execute Named Macro	Alt-X
Execute Settings	Alt-6
Execute Window	Alt-=
Exit	Ctrl-Z
Expand	Alt-F7
Expand Language	Alt-L Alt-E
Find	Ctrl-F
Find and Replace	Ctrl-R
Find Reverse	Ctrl-S
Frame/Move	Alt-F8
Goto	Ctrl-G
Help	Alt-H
Home	HOME key
Hyper Help	Alt-7
Insert File	Alt-I
Insert First Language Symbol	Alt-L Alt-1
Insert Line	Ctrl-L
Insert Number	Alt-3
Insert Special Character	Ctrl-Q
Insert/Overstrike	Ins
Join Line	Alt-K
Kill Window	Alt-F4
Left of Screen	Ctrl-Home
Line Drawing	Alt-Q
Mark Columns	Shift-F1
Mark Lines	F1
Menus	ESC
Name Macro	Alt-V
New File	Ctrl-N
Next Error	Alt-0

Function	Key Binding
Next Language Symbol	Alt-L Alt-N
Page Down	PgDn
Page Up	PgUp
Paste	F7
Previous Error	Alt-9
Previous Language Symbol	Alt-L Alt-P
Reformat Paragraph	Ctrl-P
Reformat Special Paragraph	Alt-P
Repeat Find	Alt-F
Repeat Find Reverse	Alt-S
Repeat Key	Alt-4
Repeat Replace	Alt-R
Reread File	Alt-N
Return	Ctrl-J
Return and Insert Line	ENTER
Review Window	Alt --
Right of Screen	Ctrl-End
Row Down	↓
Row Up	↑
Select	Alt-F6
Settings	Alt-A
Shift Left	F5
Shift Right	F6
Start of File	Ctrl-PgUp
Switch to File	Alt-G
Switch to Window	Alt-F5
Tab	TAB key
Tab Left	Shift-TAB
Transpose	Ctrl-T
Undelete Line	Ctrl-U
Undo	Alt-U
Unmark	F2
View File	Alt-O
Window 1	Ctrl-F1
Window 10	Ctrl-F10
Window 2	Ctrl-F2

Key Bindings

Function	Key Binding
Window 3	Ctrl-F3
Window 4	Ctrl-F4
Window 5	Ctrl-F5
Window 6	Ctrl-F6
Window 7	Ctrl-F7
Window 8	Ctrl-F8
Window 9	Ctrl-F9
Window Color	Alt-F10
Word Left	Ctrl- ←
Word Right	Ctrl- →
Write File	Alt-W
Yank Line	Ctrl-Y

Index

A

abort, 44
 Ada Language Reference Manual, accessing, 105
 ada main subprogram name, 93
 Ada programs, developing, 12
 Ada Utility Library, 13
 Ada Utility Library online documentation, accessing, 105
 add library link, 75
 aggressively inline, 93
 annotate assembly code with Ada, 91
 ASCII chart, 62
 augment library, 76
 Augment Options menu, 102—103
 auto tab, 79
 auto-insert, 79
 automatic backup, setting, 84
 automatic language expansion, 81
 auxiliary directory name, 101

B

backspace, 51
 backspace and wrap, 51
 backup, 84
 backup file, 44
 balance, 53
 bind macro to key, 60
 break and next line, 49
 break line, 49
 break long lines, 84
 bullet, 61
 bullet off, 62

C

case sensitive search, 79
 center line, 61
 Change Bindings menu, 87—88
 change case, 50
 change case marked lines, 50
 change case mode, 82
 change internal library name, 77
 change library comment, 77
 change library link, 75
 checkpoint frequency, 82
 close window, 66
 Color menu, 85—86
 color, setting, 85—86
 color, setting for
 error messages, 86
 help displays, 86
 help reference phrases, 86
 highlighted topic header, 86
 menus, 86
 status line, 86
 windows, 22, 86
 color/B&W, 86
 column, 85
 column left, 47
 column right, 47
 compile file, 67
 Compile menu, 67
 Compile Options menu, 90—92
 compile verbosely, 91
 compiler
 developing Ada programs, 12
 options, 90—92
 using, 12, 36—37, 67—68
 using Ada program libraries, 12—13
 compiler directory, 89
 compiler program name, 101
 connected link syntax, 103
 copy, 54

- copy & append, 56
- create new library, 76
- Create Options menu, 101—102
- create relinkable output, 93
- current Ada library, 89
- cursor movement, 6, 47—48
- cursor size, 82
- customizing ACE, 10
- cut, 55
- Cut & Append, 56
- Cut & Paste menu, 54—57
- cutting & pasting, 17—18
- cutting & pasting text, 54

D

- debugger, using, 37—40
- define macro, 59
- delete 0 length files, 84
- delete character, 51
- delete character and wrap, 51
- delete language symbol, 62
- delete line, 52
- delete macro, 60
- delete marked lines, 52
- Delete menu, 51—52
- delete to end of line, 52
- delete to next delimiter, 51
- delete word, 51
- disable floating point checks, 90
- display link operations, 94
- do command, 72
- DOS commands, entering from ACE, 25
- DOS Environment Library, 13
- DOS Environment Library online documentation, accessing, 105
- DOS shell, 73
- DOS Window, 6, 73

E

- Edit menu, 46—63
- edit options, 78—81
- edit screen, 66
- editing-related options, setting, 18—22
- enable global optimization, 91
- end of file, 48
- end of line, 48
- end of screen, 48
- Environment Options menu, 89
- error color, 86
- execute journal file, 60
- execute macro, 59
- execute named macro, 59
- execute settings, 70—71
- Execute Window, 6, 72
- exit, 44
- exiting, 2
- expand, 65
- expand language symbol, 62

F

- file, 85
- File menu, 43—45
- File Options menu, 83—84
- files, using, 15—18
- find, 53
- Find & Replace menu, 52—53
- find and replace, 53
- find reverse, 53
- frame/move, 65

G

- generate 80286 instructions, 90
- generate assembly code, 91
- generate debugging information, 91
- generate error log file, 91
- generate exception location date, 91

generate extended mode program, 93
 generate listing file, 91
 generate object code, 91
 generate relevant text only, 92
 get info, 68
 get last listing, 77
 get library info, 77
 get run info, 69
 goto, 48
 goto start marker, 48

H

help, 104
 getting, 2, 104
 help active reference phrase color, 86
 help color, 86
 Help menu, 104
 help reference phrase color, 86
 help topic header color, 86
 home, 48
 hyper help, 105

I

ignore extraneous EOFs, 83
 inhibit static initialization, 91
 Insert & Change menu, 49—50
 insert #, 82
 insert characters, 79
 insert date, 50
 insert file, 50
 insert first language symbol, 62
 insert function name, 79
 insert key name, 79
 insert line, 49
 insert number, 50
 insert special character, 62
 insert tabs, 79
 insert time, 50

insert/overstrike, 49

J

join and compress line, 52
 join line, 52
 justify, 79

K

key bindings, changing, 22—24, 87—88
 kill buffer, 85
 Kill Window, 7, 65

L

Language menu, 62—63
 Language Reference Manual, accessing, 105
 language symbol expansion, using, 27—36
 Language Window, 7, 63, 65
 left of screen, 48
 library levels to list, 99
 library link to add #1, 101
 library link to add #2, 101
 library link to add #3, 102
 library link to add #4—6, 102
 Library menu, 74
 Library Options menu, 97—103
 line drawing, 50
 line number, 85
 line size, 82
 link in software floating point, 93
 Link Options menu, 92—94
 link output file, 94
 link program, 68
 link verbosely, 94
 linker, using, 37, 68
 list all unit information, 98
 list body/subunit information, 98
 list dates/times, 99
 list dependencies, 98

Index

- list dependent units, 99
- list header information, 98
- list in reverse time order, 99
- list internal file names, 99
- list library, 74
- list link entries, 98
- list miscellaneous information, 99
- List Options menu, 98—99
- list source file name, 99
- list units dependent on, 99
- lock library, 76
- log changes, 80
- logging granularity, 83

M

- macros, 11—12
 - creating and using, 25—27
- Macros menu, 59—60
- Main menu, 10, 42
- main stack size, 94
- make file, 95
- Make Options menu, 95
- make program, 68
- mark columns, 54
- mark lines, 54
- memory available, 85
- menu color, 86
- menus
 - exiting, 9
 - selecting, 8
 - using, 8—10
- mouse scaling, 83
- moving the cursor, 47

N

- name macro, 59
- new file, creating, 43
- next error, 71
- next language symbol, 63

O

- obey pragma page directives, 92
- operating system services, 72
- options, 85
- Options menu, 78—103
- other compiler options, 92
- other link options, 94
- overstrike mode, 49

P

- page down, 48
- page up, 48
- paste, 55
- paste buffer, 85
- Paste Window, 7, 57, 65
- pasting text, 55
- perform global optimization, 93
- perform local optimization, 91, 93
- previous error, 71
- previous language symbol, 63
- print, 56
- printing marked text, 56
- produce line map, 94
- program arguments, 96
- program name, 96
- put link information, 102

R

- read bindings, 87
- read config, 88
- read macros, 60
- reformat block, 61
- reformat paragraph, 61
- reformat special paragraph, 61
- regular expression (RE) search, 79
- remove all library units, 75
- remove augment information, 103
- remove columns, 57

remove library link, 75
 remove library unit, 74
 remove local dependencies, 100
 Remove Options menu, 100
 remove unit body, 100
 remove verbosely, 100
 repeat find, 53
 repeat find reverse, 53
 repeat key, 60
 repeat replace, 53
 replace columns, 57
 reread file, 44
 retain backup across sessions, 84
 retain log across sessions, 80
 retain paste buffer across sessions, 80
 return, 48
 return and insert line, 49
 Review Window, 6, 65, 72
 right margin, 82
 right of screen, 48
 row down, 47
 row up, 47
 ruler, 62
 Run menu, 69
 Run Options menu, 96—97
 run program, 69
 running a program, 69—73
 running an Ada Program, 37

S

save standard output, 97
 save/restore, 80
 screen rows, 83
 select, 64
 set delimiters, 87
 Settings Menu, 81—83
 shift by tab stops, 81
 shift left, 55

shift right, 55
 show tab character, 79
 standard program library, 12
 start of file, 48
 starting ACE, 3
 status color, 86
 status line, 5, 84
 setting color for, 86
 setting options for display, 84—85
 Status menu, 84—85
 suppress all checks, 90
 suppress link step, 95
 suppress numeric checking, 90
 suppress optimizer warnings, 93
 suppress warning messages, 91
 swap out environment, 96
 switch to file, 44
 switch to window, 65
 system variables, 11

T

tab, 49
 tab compress/expand, 79
 tab left, 49
 tab size, 82
 task stack size, 94
 transpose, 50
 trim spaces, 79
 tutorials, 15—40
 compiling, linking, and running an Ada program,
 36—37
 customizing ACE, 18—25
 cutting & pasting, 17—18
 debugger, using, 37—40
 DOS commands, entering from ACE, 25
 editing-related options, setting, 18—22
 files and windows, using, 15—18
 key bindings, changing, 22—24
 language symbol expansion, using, 27—36
 macros, creating and using, 25—27
 preparing for, 15
 window color, setting, 22
 windows, framing, resizing, and positioning,
 24—25

Index

U

undelete line, 58
undo, 58
Undo menu, 58
unlock library, 77
unmark, 54
use continuous forms, 92
use preemptive tasking run-time, 94
use standard output, 92
use virtual scratch file, 94

V

view file, 44

W

window color, 86

window number, 84

windows, 3—4
 framing, resizing and positioning, 24—25

Windows menu, 64

windows, using, 15—18

word left, 47

Word Processing menu, 61—62

word right, 48

word wrap, 79

write bindings, 87

write config, 88

write file, 44

write macro to journal file, 60

write macros, 60

Y

yank line, 58